2020-09

CREATING UNDERWATER SOUNDS USING GENERATIVE ADVERSARIAL NETWORKS

Thiem, Nathan

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CREATING UNDERWATER SOUNDS USING GENERATIVE ADVERSARIAL NETWORKS

by

Nathan Thiem

September 2020

Thesis Advisor: James B. Michael
Co-Advisor: Marko Orescanin

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2020	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE CREATING UNDERWATER SOUNDS USING GENERATIVE ADVERSARIAL NETWORKS	**5. FUNDING NUMBERS**
6. AUTHOR(S) Nathan Thiem	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.	**12b. DISTRIBUTION CODE** A

13. ABSTRACT (maximum 200 words)

Current generative adversarial network (GAN) synthesized audio is full of artifacts that can cause it to sound unnatural or machine-like. This thesis proposes a GAN architecture, TangGAN, which reduces artifacts in generated audio through the introduction of anti-aliasing filters in the discriminator and interpolation methods in the generator to correct improper sampling that occurs in convolutional layers. TangGAN's performance was evaluated on spoken digit, pure tonal, and underwater acoustic datasets through comparison with a current state-of-the-art audio GAN, WaveGAN. TangGAN showed improvement over WaveGAN as measured by the standard audio GAN metric of Inception score, as well as by three new metrics proposed in this work to quantify artifacts in GAN-produced audio: speech-to-reverberation modulation energy ratio; total harmonic distortion; and signal-to-noise ratio. The reduction of artifacts in GAN-generated audio is a necessary step to the fitness for use of these methods in both the commercial and DOD environments.

14. SUBJECT TERMS generative adversarial network, GAN, acoustics, artificial intelligence, AI, Machine Learning, AI/ML, underwater acoustics, WaveGAN	**15. NUMBER OF PAGES** 95
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

CREATING UNDERWATER SOUNDS USING GENERATIVE ADVERSARIAL NETWORKS

Nathan Thiem
Lieutenant, United States Navy
BS, Drexel University, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2020

Approved by: James B. Michael
Advisor

Marko Orescanin
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Current generative adversarial network (GAN) synthesized audio is full of artifacts that can cause it to sound unnatural or machine-like. This thesis proposes a GAN architecture, TangGAN, which reduces artifacts in generated audio through the introduction of anti-aliasing filters in the discriminator and interpolation methods in the generator to correct improper sampling that occurs in convolutional layers. TangGAN's performance was evaluated on spoken digit, pure tonal, and underwater acoustic datasets through comparison with a current state-of-the-art audio GAN, WaveGAN. TangGAN showed improvement over WaveGAN as measured by the standard audio GAN metric of Inception score, as well as by three new metrics proposed in this work to quantify artifacts in GAN-produced audio: speech-to-reverberation modulation energy ratio; total harmonic distortion; and signal-to-noise ratio. The reduction of artifacts in GAN-generated audio is a necessary step to the fitness for use of these methods in both the commercial and DOD environments.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AIS	Automatic Identification System
CAE	contractive auto-encoder
CGAN	conditional generative adversarial net
CIFAR	Canadian Institute for Advanced Research
DBN	deep belief network
DCGAN	deep convolutional adversarial network
DOD	Department of Defense
FID	Fréchet Inception distance
FIR	finite impulse response
GAN	generative adversarial network
GSN	generative stochastic network
GPU	graphics processing unit
IF	instantaneous frequency
IS	Inception score
LOFAR	low frequency analysis and recording
LReLU	leaky rectified linear unit function
MNIST	Modified NIST (National Institute of Standards and Technology)
MOS	mean opinion score
NDB	number of statistically different bins

NIST National Institute of Standards and Technology

NPS Naval Postgraduate School

NUWC Naval Undersea Warfare Center

NUWDC Naval Undersea Warfighting Development Center

PESQ perceptual evaluation of speech quality

ReLU rectified linear unit

SC09 Speech Commands Zero Through Nine

SNR signal-to-noise ratio

SRMR speech-to-reverberation modulation energy ratio

STFT short-time fourier transform

THD total harmonic distortion

USN United States Navy

WGAN-GP Wasserstein generative adversarial network – gradient penalty

Acknowledgments

Although this is one of the first parts of this thesis, it is the one I wrote last as it will be difficult to fit succinctly express the amount of gratitude I owe to the people who helped make this thesis possible. I would like to start by thanking my advisors, Professors Michael and Orescanin. Professor Michael, without your guidance and mentorship I would have never been able to focus the content of this thesis or had the opportunity to work with most of the other people on this list. As pointed out in the book, *The Innovator's Way*, social capital is the accumulated trust among others that opens space for action, and as a graduate student navigating a new environment with no social capital, I was glad you shared yours with me. Professor Orescanin, I'm honored to be your first thesis student at the Naval Postgraduate School (NPS). There are very few artificial intelligence/machine learning experts and very few acoustics experts in the world. I consider myself incredibly fortunate to have had an expert in both as an advisor. Plus, I enjoyed our conversations on science fiction, the military, and family life when we got distracted from work. To Provost Robert Dell, I am grateful for the funding support in allowing me to attend the 2020 Naval Applications of Machine Learning conference in San Diego to present a preliminary version of this work. I am indebted to Dr. Peter Denning whose passionate belief that innovation is more than invention and that the hard part of innovation is getting others to adopt new practices, inspired me to reach out both within and outside of the NPS community to talk about my work. None of this work would have been happened without the support of the High Performance Computing Center at NPS; friendly and responsive, they are a professional group with whom more faculty, staff, and students should work, especially as the U.S. Navy turns to high-performance computing to compete in the twenty-first century. I have to acknowledge the support I received from the Naval Undersea Warfare Center (NUWC) and Naval Undersea Warfighting Development Center (NUWDC) for pointing me in the direction of GANs for study. I also have to thank my fellow submariners here at NPS who were never afraid to remind me that even the worst day of thesis work was better than spending time on a submarine. I look forward to seeing all of you again out in the fleet. Lastly, and most importantly, I must thank my lovely wife, Marixi. I know it must have been frustrating to watch me pounding away at a keyboard for countless hours at home as we spent the last six months learning from home, but without your patience, understanding, and kindness none of this would have been possible.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Teaching computers how to create original data that does not appear to be computer generated is a challenging problem. Not all data types are equally challenging for computers to learn to generate convincingly. Audio data is one of the most challenging, not because it is inherently more complex than photo-quality images, coherent text, or video, but because the human auditory system is sensitive to any audible artifacts a computer introduces in generation. These artifacts are more noticeable to humans than artifacts in other data types. They cause generated audio samples to be perceived as distorted and choppy causing human listeners to recognize audio as machine-like and unnatural (full of artifacts). Learning how to prevent artifacts is a major barrier to computer-generated audio.

Recent advances in machine learning like generative adversarial networks (GANs) have contributed to this pursuit. This thesis develops a GAN architecture, TangGAN, that attempts to generate natural-sounding audio by reducing the number of artifacts present. TangGAN is the first audio GAN architecture that formally addresses the issue of aliasing artifacts introduced by convolutional layers. Aliasing artifacts are the improper manifestation of high frequency sound features in a lower frequency feature space caused by improper downsampling. The artifacts are addressed through the introduction of decimation layers in the discriminator and interpolation layers in the generator. This thesis explores possible architectures for TangGAN, benchmarks TangGAN's performance against another successful audio GAN, and evaluates TangGAN's ability to reduce artifacts in generated audio trained on three diverse audio datasets.

1.1 Motivation

Submarines are one of the most survivable and lethal platforms in the entire United States' arsenal. Their unique capabilities, however, depend entirely on their stealth. Without this stealth a submarine becomes vulnerable. There are many ways a submarine can be counterdetected, but one that is especially concerning is the risk of interception of active communications. The risk of communications is not just theoretical. During World War II,

German U-boats were able to effectively attack Allied shipping through the use of the use of the "wolf pack," a tactic where a group of submarines in direct communication coordinated a simultaneous attack on a convoy of Allied ships. This tactic was so effective that British Prime Minister Winston Churchill said, "The only peril that ever really frightened me during the war was the U-boat peril." To win the Battle of the Atlantic, the Allies turned the wolf pack against the U-boats by breaking the German military's Enigma Code. After breaking the code, the Allies could intercept and understand communications between U-boats and their headquarters. The U-boats' loss of stealth stealth helped the Allies ensure that over 80% of all U-boaters never returned to port [1].

This risk of communications interception to submarines is not just a problem of the past. Almost every submariner today has a personal story about how a mission has been impacted by communications. One way to overcome the risk is to simply not communicate. As we move into the twenty-first century, forgoing communications becomes an even less tenable option. Without communications, high-tech capabilities such as controlling underwater drones from a submerged submarine are not possible. Even low-tech, but devastating tactics necessary in a great power war like the wolf pack are not possible. Clearly, the inability to communicate limits submarines' lethality for security.

This tradeoff between lethality and security does not need to limit the submarine community any longer. One way to make communications immune to counterdetection is to embed them inside natural underwater sounds. By disguising communications as natural sounds, it will be impossible for an adversary to tell what sounds are communications and what sounds are part of the environment. Through the development of TangGAN, this thesis takes the first steps towards creating natural-sounding underwater sounds. GAN-generated sounds give an advantage over using pre-recorded sounds as a carrier signal for communications because GAN-generated sounds are unique each time they are created. This makes it impossible for the adversary to identify underwater sounds used for communication based on previously having heard the carrier signal. Using a GAN to generate the carrier signals ensures signals that are appropriate signals that are appropriate for the underwater environment; that is GANs can be used to create context-aware carrier signals. Giving submarines the ability to communicate without being counterdetected will enable them to become an even more capable force without sacrificing the stealth that makes them so valuable.

1.2 Thesis Organization

There are six additional chapters in this thesis. Chapter 2 discusses previous work that TangGAN builds on. This chapter covers GAN definition and basic architecture, a survey of previously published audio GANs, and analysis of why anti-aliasing is necessary for any architecture, like GANs, that use modern convolutional layers. Chapter 3 describes the three datasets used for GAN training, *Speech Commands 0 through 9* (SC09), the pure tone dataset, and the underwater acoustic datasets. Chapter 4 outlines the TangGAN architecture, specifically how anti-aliasing and interpolation technologies are incorporated into a GAN and why these technologies were expected to reduce artifacts in generated audio. Chapter 5 outlines the experiments and evaluation metrics TangGAN was subjected to. This chapter also explains why three new evaluation metrics, speech-to-reverberation modulation energy ratio (SRMR), total harmonic distortion (THD), and signal-to-noise ratio (SNR), were introduced to evaluate audio quality through quantifying artifacts. Chapter 6 presents the results of the preceding chapter's experiments and offers a discussion of those results. Finally, Chapter 7 offers a conclusion and most importantly suggestions for further research in the field of audio GAN research.

This code written for this thesis can be found in the repository online [2]. The data used for training and the best network weights used for generating audio also can be found online [3].

1.3 Findings and Contributions

This thesis makes contributions to audio GAN architecture and evaluation. TangGAN formally addresses aliasing caused by improper sampling in convolutional layers in GAN discriminators and generators. TangGAN deals with aliasing through the addition of decimation layers in the discriminator and interpolation layers in the generator. Addressing aliasing allows TangGAN to produce sound with fewer artifacts than the state-of-the-art WaveGAN. Since current metrics for evaluating GAN-produced audio do not quantify artifacts, three new metrics are introduced: speech-to-reverberation modulation energy ratio (SRMR); total harmonic distortion (THD); and signal-to-noise ratio (SNR). Contributions to audio GAN architecture and evaluation are necessary before GANs can be adopted for Department of Defense (DOD) or commercial use.

TangGAN provides better performance over WaveGAN as evaluated by traditional and

novel metrics on three datasets. On the SC09 dataset, TangGAN showed improvements in Inception score (IS) and SRMR demonstrating that TangGAN improved human speech intelligibility over WaveGAN. TangGAN improved THD and SNR on the pure tone dataset showing that TangGAN produced fewer artifacts than WaveGAN. Finally, qualitative analysis performed on spectrograms of GAN-produced ship sound samples showed TangGAN prevented the introduction of checkerboard-style artifacts unlike WaveGAN. TangGAN introduces a new audio GAN architecture that improves on the current state-of-the-art as evaluated both by traditional and new metrics.

CHAPTER 2:
Previous Work

2.1 Generative Adversarial Networks

2.1.1 The Original GAN

The field of deep learning has blossomed over the last decade with developing neural network model architectures for discriminative learning tasks. These discriminative models take, "high dimensional, rich sensory inputs," and learn a mapping function between a model distribution and a data distribution. The most successful neural network architectures for discriminative tasks utilize recent advancements such as dropout and piecewise linear units. However, these newly formed architectures have not excelled at producing high-quality generative models. The first Generative Adversarial Networks (GANs) were introduced in 2014 by Goodfellow et al. [4]. The authors suggested that challenges generative models face are due to the difficulty in managing piecewise linear units and probabilistic computations in a generative task. To overcome these challenges the authors proposed a new architecture.

The original GAN architecture consists of three main parts: a generator, a discriminator, and a loss function. The idea behind a GAN is that a neural network model known as a generator draws a sample from a random distribution and learns how to map that latent vector into data distribution space. The output of this mapping is known as "fake" data. The role of the discriminator, a second neural network model, is to determine whether the sample from the generator is drawn from the real data distribution or the fake data. This setup forms a minimax game. This game is described via the loss function that is optimized in the process of training. The goal of the generator is to maximize the number of mistakes the discriminator makes in determining whether the sample is generated. The goal of the discriminator is to minimize the number of mistakes it makes in the same determination. In [4], the GAN team makes a comparison of GANs to counterfeit currency. The generator can be viewed as a team of counterfeiters trying to make and use fake currency without detection while the discriminator can be viewed as a law enforcement team trying to determine the difference between counterfeit and real currency. As the law enforcement

team gets better at determining counterfeit and real currency, the counterfeiters must become better at producing counterfeit currency that looks real.

Mathematically, [4] describes the generator as follows. A goal of generator is to learn a probability distribution, p_g over data $\boldsymbol{x}$. The generator is initialized with a random noise vector z, a latent vector, and learns to map z to a distribution, p_g. The generator can therefore be described as $p_g(z) = G(z; \theta_g)$, where G is a neural network architecture with parameters θ_g.

Similarly, the goal of the discriminator in [4], D, is described as learning to estimate the probability of $\boldsymbol{x}$ being drawn from real data instead of p_g. This is expressed as $D(\boldsymbol{x}) = D(\boldsymbol{x}; \theta_x)$, where θ_x are the parameters of the neural network, D.

The loss function, or value function $V(D, G)$, is the well-known minimax. In the process of optimization, D is trained to maximize the probability of assigning the correct real/generated label to $\boldsymbol{x}$ and G is trained to minimize the probability that that the discriminator assigns the proper label to $G(z)$. The competing training goals are combined in the following function:

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p_{\text{data}(x)}}(\log D(\boldsymbol{x})) + \mathbb{E}_{z \sim p_{z(z)}}(\log(1 - D(G(z)))).$$

Goodfellow et al. [4] offered several insights into GAN training. The first insight is that a GAN is not an optimization problem. Rather, it is a minimax game that can stop training at any saddle point that maximizes the generator's ability to trick the discriminator while minimizing the generator's chances of misclassifying samples. The only time the GAN finds the absolute minimax point is when the generated data is identical to real data or, $p_g = p_{\text{data}}$. All other points are local minimax points. The second insight is that the probability function of the generator will converge to the probability function of the real data as long as the generator and discriminator have enough capacity, the discriminator is allowed to reach its optimum for a given generator, and the generator is updated to improve the loss function. However, the GAN authors point out that the generators and discriminators rarely have enough capacity (number of model parameters), so this convergence is not guaranteed to be true. Nevertheless, the authors demonstrate through experimentation that the proposed GAN architecture and overall minimax game approach show promise on generative tasks.

To demonstrate viability of the GAN approach, the authors trained their model using multiple datasets including Modified National Institute of Standards and Technology (MNIST) [5], the Toronto Face Database [6], and Canadian Institute for Advanced Research-10 (CIFAR-10) [7]. MNIST was developed by researchers in [5] through the adaptation of NIST datasets depicting handwritten digits by centering the digits in the images and normalizing pixel values. MNIST contains $60,000$ images of digits that are 28 by 28 pixels in size. The Toronto Face Database is a private database created at the University of Toronto containing grayscale versions of human faces [6]. CIFAR-10 is a collection of $60,000$ 32 by 32 pixel color images in ten classes [7].

The authors of the original GAN paper experimented with convolution layers in the discriminator and deconvolutional layers in the generator, as well as fully connected neural networks for both models. In the generator they used rectified linear units and sigmoid activation functions while in the discriminator they used dropout and maxout activations. The neural networks were trained on the training datasets as described above and performance was qualitatively verified by inspection of the output. The generated images were visually compared with their nearest neighbors to ensure the generator did not overfit or memorize the training dataset. Quantitatively, the performance of the GAN was measured by fitting a Gaussian Parzen window to generated data and comparing to validation data to generate Parzen window-based log-likelihood estimates. The Parzen window-based log-likelihood estimates for the GAN were compared to three non-GAN based generative architectures: deep belief network (DBN) [8], stacked contractive auto-encoder (CAE) [8], and deep generative stochastic network (GSN) [9]. The GAN outscored the other methods for MNIST and higher than Deep GSN and DBN on the Toronto Face Database. With these scores, the GAN team concluded that GANs are competitive with other high-quality state-of-the-art generators.

2.1.2 Progressively Growing GANs

One of the most influential papers on GANs was published in 2018 [10]. Written by a team of researchers from NVIDIA, this paper described a new GAN architecture, progressively growing GANs, capable of generating high-quality images at reduced training time relative to non-progressively growing GANs. Progressively growing a GAN works by growing the size of a GAN model for the discriminator and the generator as the GAN trains. For

example, if the ultimate output of a GAN were 16 by 16 pixel images, a progressively growing GAN may start by training a GAN that has a generator that outputs a two by two pixel image and a discriminator that takes in a two by two pixel image. After the model reaches convergence in this architecture configuration another layer is introduced to both the discriminator and generator such that generator output grows to four by four pixel images and the discriminator input scales to classify four by four pixel images. This process is repeated until the desired generation size of 16 by 16 pixels is achieved.

The NIVDIA team in [10] introduced several new improvements to GAN training methodology. First, the NVIDIA team found that slowly introducing layers when growing a GAN helped prevent a "shock" to lower resolution layers that had already been trained. Preventing shock allowed the lower resolution layers to build off of what they learned when the progressively growing GAN was smaller instead of having to relearn when a higher resolution layer was introduced. The NVIDIA team slowly introduced a layer by having a weighting factor, α, that combined the output of a new layer with the output of the proceeding lower resolution layer; this is equivalent to exponential smoothing. The weighting factor is gradually increased as the GAN is trained until the new layer's output is the only output of the system. A diagram from [10] can be seen in Figure 2.1.

To increase the diversity of generated outputs features were introduced into the discriminator by introducing a minibatch layer near the end of the discriminator that learns an array of statistics from each minibatch. This minibatch discrimination method is based off of the work published in [11].

Additionally, normalization layers that had been introduced into many GAN architectures after their suggestion by [12] were removed from the progressively growing GAN architecture. Batch normalization layers were developed by [12] to allow deep neural networks like GANs to train more quickly with higher learning rates. Adopting batch normalization layers allowed [12] to create an Inception Net based image classifier that trained with an order of magnitude less steps than the original Inception Net. The NVIDIA team in [10] designed their architecture for their GAN to omit batch normalization layers through adopting a new technique termed "equalized learning rate." The equalized learning rate initializes weights randomly between zero and one and adjusts the rates during runtime by the relation $\hat{w}_i = w_i/c$, where $\hat{w}_i$ is the new weight, w_i is the old weight, and c is the per

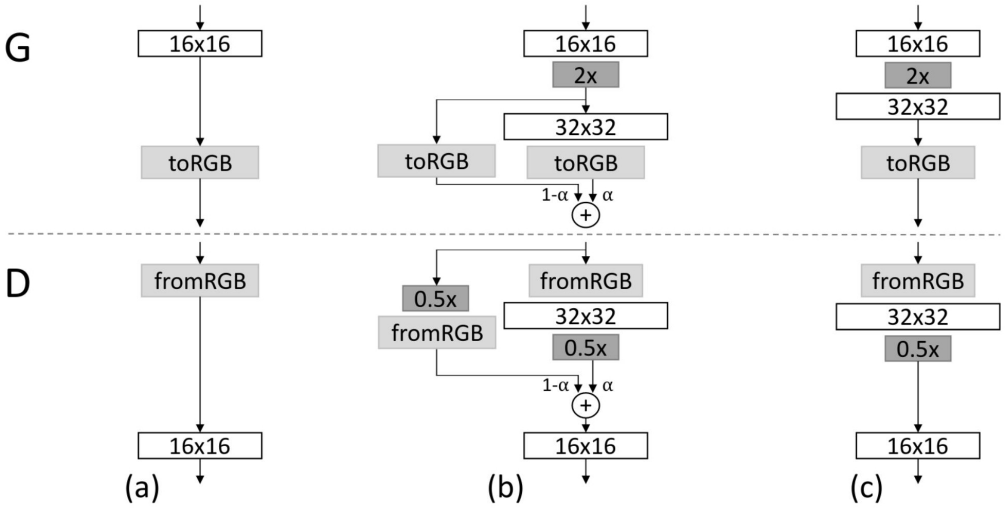

Figure 2.1. This diagram from [10] illustrates the shock prevention approach of introducing a new layer into a progressively growing GAN slowly. The portion above the line marked G represents the generator architecture while the portion below the line marked D represents the discriminator architecture. The architecture of a 16 by 16 generator and discriminator are in (a). After new layers are introduced to expand the architectures to a 32 by 32 generator and discriminator, the new layers' outputs are combined with the old layers' outputs by multiplying by a factor of α and $1 - \alpha$, respectively, as seen in (b). As the GAN trains, alpha is gradually increased so that the new layers contribute more and more to the generator and discriminator. Eventually, as seen in (c), α reaches one and the new layer is no longer bypassed.

layer normalization constant from He's initializer [13]. Adjusting weights in this manner allowed the weights to be adjusted by the same relative amount regardless of the scale. This scaling allowed for higher learning rates because weights with large scales are no longer adjusted disproportionately to weights with smaller scales. Furthermore, pixelwise vector normalizations were introduced in the generator. This addition prevented the generator from raising the magnitudes of features uncontrollably. The NVIDIA team, however, noted that pixelwise vector normalization had little effect on their GAN results.

The progressively growing GAN architecture was tested on several datasets including CIFAR-10 [7], LSUN [14], and CelebA-HQ. The most impressive results were obtained using CelebA-HQ created by the NVIDIA team from CelebA [15]. CelebA is a dataset con-

sisting of photographs of celebrities. There are ten thousand identities, or people in CelebA, each of which has 20 images for a total of two hundred thousand images. The images in CelebA have a wide range of resolutions, number of people in an image, and orientation of faces. The NVIDIA team in [10] created CelebA-HQ through an image processing pipeline on all CelebA images. The pipeline removed artifacts, increased resolution, mirror padded, applied a depth-of-field effect to blur the background, cropped each image on the main face, and resampled each image so that each image had a 1024 by 1024 pixel resolution. While this process was performed on all two hundred thousand images of CelebA, only the best thirty thousand images were kept and saved as CelebA-HQ.

The results from the progressively growing GAN as applied to Celeba-HQ are impressive considering the size and the quality of the produced images. The images produced in [10] make the progressively growing GAN one of the first GANs to generate high-resolution data. While the original GAN [4] was generating CIFAR-10 like images of 32 by 32 pixel resolution, [10] generates images with a 1024 by 1024 pixel resolution. The progressively growing GAN generates over a thousand times more pixels than the original GAN. Additionally, the generated images of people are of photorealistic quality. Examples of faces generated by a progressively grown GAN in [10] can be seen in Figure 2.2. The quality of produced images is so high that images produced by StyleGAN, a version of the progressively grown GAN upgraded by the same team at NVIDIA [16], are used on websites such as whichfaceisreal.com which challenge people to identify which of two images is real and which was generated by StyleGAN [17]. Distinguishing between the two is not as easy as it would seem.

The quality of the progressively growing GAN can be measured through an experiment that the NVIDIA team performed on CIFAR-10 [10]. A progressively growing GAN was trained on the CIFAR-10 dataset which generated images with a 32 by 32 pixel resolution. Unlike the faces in CelebA-HQ, CIFAR-10 is smaller in resolution and is a labeled dataset [7]. This allowed the NVIDIA team to calculate an Inception score. Inception score measures the quality and diversity of generated samples by classifying generated samples with a pretrained inception classifier. Inception score is discussed in further detail in the WaveGAN subsection of this chapter. In this case, a GAN trained on CIFAR-10 would have a maximum score of 10, where 10 is related to the number of unique labels in the dataset. The progressively growing GAN on CIFAR-10 had an Inception score of 8.80. According to [10]

Figure 2.2. These are all computer-simulated faces, not photographs of real people. Each face was generated by the progressively growing GAN in [10].

this beat the previously state-of-the-art unsupervised GAN Inception score on CIFAR-10 of 7.90 achieved by Splitting GAN [18].

Progressively growing GAN trained on a 1024 by 1024 pixel per image dataset like Celeba-HQ trains 5.4 times faster than a non-progressive version [10]. Scalability is an important factor in selecting a training approach. Training for the GAN was conducted over a period of four days of training on eight NVIDIA Tesla V100 GPUs to achieve the reported results.

2.2 GANs for Audio Synthesis

2.2.1 Expansion of Restricted Samples based on GANs

Fan Liu, Qingzeng Song, and Guanghao Jin of the Tianjin Polytechnic University use GANs to generate spectrograms of underwater sounds in [19]. Interestingly, from a naval perspective, the purpose of their work is to produce additional examples of low-frequency analysis and recording (LOFAR) spectrograms to expand an "original dataset in [an] underwater acoustic signal case, thereby overcoming the problem of insufficient data samples for classification and identification tasks" [19]. That is, the team at Tianjin is trying to artificially create additional LOFAR spectrograms from hard to collect sonar data sources to improve

the quality of sonar classifiers.

From a technical standpoint, [19] is interesting because it uses a GAN to generate underwater sound spectrograms. To generate these spectrograms, [19] starts with .wav audio files. The audio files are preprocessed into a LOFAR spectrogram by performing a short-time Fourier transform (STFT). In order to focus on specific targets [19] LOFAR spectrograms are generated within the bandwidth of the frequency range governed by sampling rate in order to develop a training dataset for their GAN. These spectrograms are further converted into gray values with data energy interpolation. Finally, to increase training speed, only one LOFAR spectrogram was chosen for every 100 seconds and the resolution of the spectrograms was compressed from 512 by 512 pixels per spectrogram to 64 by 64 pixels per spectrogram.

After the data preprocessing, the dataset is used to train [19]'s GAN. This GAN consists of two parts, a generator and a discriminator. The generator takes a latent vector of size 77 and grows it to a LOFAR spectrogram of size 64 by 64. The GAN grows the latent vector into the data for a full LOFAR spectrogram by applying two fully connected layers followed by three up-convolution layers. After each layer, except the final up-convolution layer, a leaky rectified linear unit function (LReLU) is applied and layers are batch normalized. The up-convolution layers use a kernel size of four by four and a stride of two. The final step to produce a LOFAR spectrogram is to apply data processing and gray scale mapping to output a spectrogram from the generator's output.

The discriminator architecture in [19] is composed of similar layers to the generator but reversely ordered. A LOFAR spectrogram is presented to the discriminator to decide whether the image is real. The architecture of the discriminator consists of three two-dimensional convolution layers followed by two fully connected layers. Like for the generator, the convolution layers are built using four by four kernel filters with a stride of two. After the last two convolution layers and the first fully connected layer, a LReLU function and batch normalization are applied. The output of the last fully connected layer is a single value which has a sigmoid activation function applied.

The generator and discriminator are coupled via the Conditional Generative Adversarial Net (CGAN) loss function proposed in [20]. The team in [19] chose the CGAN loss function in order to control the class of the generated sounds based on a label that can be given to the

generator and discriminator.

To evaluate the quality of their proposed GAN, Liu, Song, and Jin performed three experiments on a classifier based on the AlexNET neural network architecture [21]. In the first experiment, the classifier was trained and validated exclusively on real data. This experiment set a baseline on the effectiveness of the classifier with an unexpanded dataset, or the dataset not augmented with fake images. The second experiment trained on real data but was validated on GAN generated data (fake data) to show that the generated data produced spectrograms were classifiable according to their label. The final experiment trained and validated on a dataset combining real and generated data to test the hypothesis that LOFAR spectrogram classifier accuracy could be increased by adding generated data to the classifier's training data. Liu, Song, and Jin were able to improve the accuracy of their classifier on real data from 75.7% in their first experiment to 91.2% in their last experiment.

It is important to note that the GAN in [19] does not generate audio but 64 by 64 pixel LOFAR spectrograms. To invert a spectrogram into raw audio, an inverse Fourier transform is required along with assumptions about the sound's phase information. While the method used in [19] for preprocessing raw .wav files by applying a STFT preserves the frequency magnitudes of the training data, it discards the phase information. To produce realistic audio the frequency and phase information of a sound must both be preserved.

2.2.2 WaveGAN

WaveGAN, by Chris Donahue, Julian McAuley, and Miller Puckette of the University of California, San Diego, is the first application of GANs to unsupervised audio generation [22]. Like every GAN, WaveGAN consists of a generator and a discriminator combined via a loss function. WaveGAN architecture is based on the architecture of deep convolutional generative adversarial networks (DCGAN) [23]. DCGAN is a very popular image generating GAN. WaveGAN made substantial modifications to DCGAN architecture in order to adapt it to sound generation. One of the most striking differences is that WaveGAN dispenses the DCGAN loss function and replaces it with the Wasserstein generative adversarial network - gradient penalty (WGAN-GP) loss function [24].

The generator architecture in WaveGAN consists of one fully connected, or dense, layer that takes in a latent vector of length 100 and then through five one-dimensional transpose

convolutional layers grows the latent vector into an audio sample 16, 384 time steps long. After the dense and all but the last transpose convolutions, a rectified linear unit (ReLU) function is applied. After the last transpose convolution a hyperbolic tangent function is applied before a generated sound is output. Each transpose convolution layer has a stride of four with no dilation, meaning the input to each transpose convolution grows in size by a factor of four as the transpose convolution is applied. The filter size of each convolution is 25, all in one dimension, a considerable increase to the one-dimensional size of five from the five by five two-dimensional DCGAN filter. The longer filter was adopted because typical feature sizes in audio are larger than the typical feature size in images. Table 2.1 describes WaveGAN's generator.

Table 2.1. WaveGAN generator architecture.

Layer	Kernel Size	Stride
Input	n/a	n/a
Dense	n/a	n/a
Reshape	n/a	n/a
ReLu	n/a	n/a
1-D Trans Conv	25	4
ReLU	n/a	n/a
1-D Trans Conv	25	4
ReLU	n/a	n/a
1-D Trans Conv	25	4
ReLU	n/a	n/a
1-D Trans Conv	25	4
ReLU	n/a	n/a
1-D Trans Conv	25	4
Tanh	n/a	n/a

WaveGAN's discriminator architecture accepts as an input an audio sample of length 16, 384 and compresses it into a single value that estimates whether the sound was real or generated. WaveGAN achieves this via five one-dimensional convolution layers and one dense layer. A table outlining WaveGAN's discriminator architecture can be seen in Table 2.2. Like their transpose convolution counterparts in the generator, each convolution layer in the discriminator has a stride of four and a filter length of 25. Each convolution layer is followed by a LReLU function. A major deviation from DCGAN is the introduction of the

phase shuffle layer after all but the last convolution layers. The transpose convolution layers in WaveGAN create artifacts at regular intervals in the created sounds, which always occur at the same phase. This allows the discriminator to trivially identify generated sounds by these artifacts and prevents the generator from learning. WaveGAN solves this problem through phase shuffle layers. Phase shuffle works by randomly shifting all the samples points in a sound sample to the left or right by zero, one, or two time steps. Any samples that would fall outside of the sample's time domain are wrapped back around to the beginning or end of the sample. The shifts prevent artifacts from always appearing at the same phase while having little effect on the discriminator's ability to recognize actual features because the time shifts are small in size compared to the sample's time domain.

Table 2.2. WaveGAN discriminator architecture.

Layer	Kernel Size	Stride
input	n/a	n/a
1-D Conv	25	4
LReLU	n/a	n/a
Phase Shuffle	n/a	n/a
1-D Conv	25	4
LReLU	n/a	n/a
Phase Shuffle	n/a	n/a
1-D Conv	25	4
LReLU	n/a	n/a
Phase Shuffle	n/a	n/a
1-D Conv	25	4
LReLU	n/a	n/a
Phase Shuffle	n/a	n/a
1-D Conv	25	4
LReLU	n/a	n/a
Reshape	n/a	n/a
Dense	n/a	n/a

The novelty of WaveGAN's architecture is that it operates in the time domain. This means that WaveGAN trains and produces raw digital audio unlike [19], which operates in the frequency domain (i.e., ultimately the image domain) by training and producing audio spectrograms. Spectrograms are merely images of a signal spectrum evolution over time.

WaveGAN was extensively evaluated through experimentation. In order to test the validity of generating sounds in the time domain vice frequency domain another GAN called SpecGAN was created in [22] which operates in the frequency domain. SpecGAN works by preprocessing a sound sample into a 128 by 128 spectrogram through performing a STFT and scaling. These produced spectrograms are then used to train SpecGAN. SpecGAN has a nearly unmodified DCGAN architecture. When generating, SpecGAN produces spectrograms. To invert spectrograms into audio the authors utilized an iterative Griffin-Lim algorithm, which enables them to reconstruct the audio phase which was discarded in the process of making spectrograms.

WaveGAN and SpecGAN were each trained on five different datasets, drum sound effects, bird vocalizations, piano samples, large vocab speech samples, and the *Speech Command Zero Through Nine* (SC09) dataset. The SC09 dataset is the most significant of these because all of the evaluation of WaveGAN was performed after training on SC09. SC09 is a subset of the *Speech Commands* dataset [25]. SC09 consists of over $18,000$ instances of different speakers saying the words for the digits zero through nine in an uncontrolled environment. SC09 was chosen because: each sample is one second long sampled at 16 kHz making WaveGAN's size and training times manageable; generated digits can be easily validated by a human listeners; the variation in features is diverse both between digits and individual samples of the same digit; and the dataset is reminiscent of the MNIST dataset.

WaveGAN was evaluated via three metrics. The first metric was Inception score. Inception score uses a pretrained Inception classifier to measure the diversity and intelligibility of generated samples. The implementation of Inception score in [22] is as follows: An Inception classifier is trained on spectrograms created from the original SC09 dataset. $50,000$ samples are generated by WaveGAN. The generated samples are then changed into spectrograms. The spectrograms are then given to the already trained classifier. The Inception score is then calculated for each digit, x, for each sample, y, via the relation

$$e^{\mathbb{E}_x D_{\mathrm{KL}}(P(y|x)||P(y))},$$

where $\mathbb{E}_x$ is the expected probability distribution of x, D_{KL} is the Kullbeck-Leibler Divergence, and P is a probability distribution function. Since the Inception classifier and WaveGAN were both trained on the SC09 dataset which has ten categories, the maximum

possible Inception score is ten. The highest Inception score that WaveGAN achieved was 4.12 ± 0.03 which is less than the maximum Inception score of SpecGAN of 6.03 ± 0.04. The authors note that since SpecGAN generates spectrograms, not raw audio like WaveGAN, SpecGAN may have a "representational advantage" in the Inception score because the Inception classifier is trained to recognize features in spectrograms, not raw audio. Interestingly, the Inception score for the training SC09 dataset does not have a perfect score and only achieves 9.18 ± 0.04.

The second evaluation metric used was a nearest neighbor comparison. The nearest neighbor comparison was chosen to ensure that two potential cases where a high Inception score falsely represents a successful GAN were not present. High Inception scores can be achieved if a GAN either outputs the sounds it was given to train on or it learns only a single example of each class with a uniform probability. The first case is undesirable because it means the GAN is not generating new data, and the second case is undesirable because it means the GAN has not learned how to make diverse data. The nearest neighbor comparison consists of two measurements $|D|_{\text{self}}$ and $|D|_{\text{train}}$. Both measurements were taken on spectrogram representations of sound samples. $|D|_{\text{self}}$ is the average Euclidean distance between $1,000$ examples and their nearest neighbors in the set other than themselves while $|D|_{\text{train}}$ is the average Euclidean distance between samples and their nearest neighbors in the training dataset. The lowest possible $|D|_{\text{self}}$ value of zero would indicate that the GAN only generated a single example of each sound, while the lowest possible $|D|_{\text{train}}$ value of zero would indicate that the GAN is exactly replicating the training data. The nearest neighbor comparison on both WaveGAN and SpecGAN indicated that both GANs generated datasets that were diverse and distinct from the training data.

Finally, WaveGAN was qualitatively evaluated through a listening experiment with human subjects. Samples from WaveGAN and SpecGAN were evaluated by human listeners on Amazon Mechanical Turk. Listeners were presented with a batch of ten random sounds from the SC09 training set, WaveGAN, or SpecGAN. The listeners were then asked to identify the digit they heard for each of the ten samples then rate the entire batch's sound quality, ease of intelligibility, and speaker diversity. SpecGAN achieved slightly higher label accuracy than WaveGAN, but slightly lower sound quality and speaker diversity ratings. SpecGAN and WavGAN achieved the same ease of intelligibility ratings.

Ultimately, the creators of WaveGAN concluded that it is feasible to generate sounds with a GAN in both the time and frequency domains. GANs have an additional advantage over other generative models of being able to generate hours of original audio in only a few seconds, something autoregressive models struggle with.

One problem unrecognized by the authors of [22] is that the phase shuffle layer in WaveGAN only prevents artifacts from occurring at the same phase in generated samples, it does not prevent artifacts from being created. We believe these artifacts are created by the aliasing nature of the convolutional layers used in both the generator and the discriminator.

2.2.3 GANSynth

Given the commercial nature of audio, Google has continuously invested in audio GAN technology and has further developed the process of generating sounds using GANs [26]. The GANSynth team, which includes Chris Donahue, the primary author of [22], focused on generating sounds using the NSynth dataset. The NSynth dataset is a collection of $300,000$ four-second musical notes sampled at 16 kHz from over $1,000$ musical instruments. The GANSynth team chose the NSynth dataset because of its diverse sampling of musical pitches and timbres while still being highly structured.

To create better sounding audio from a GAN, the GANSynth team attempts to deal with audio artifacts by unwrapping phases. This helps alleviate artifacts that are created when a waveform frequency is not in phase with the convolutional strides used in a GAN. The process used to unwrap the strides is termed instantaneous frequency (IF). To generate sounds, spectral representations of time series data are created by using STFTs. Furthermore, the magnitude is log compressed and normalized while the phase of the STFT is unwrapped to apply IF. After pre-processing this data is passed to the GAN for training.

The GANSynth team created three GAN variants that incorporate IF for testing. The first variant, IF GAN incorporates the IF method of data preprocessing and phase unwrapping. The second variant, Phase GAN, is similar to IF GAN except it omits phase unwrapping. The third GAN, IF-MEL GAN, creates spectrograms with mel scale axes during preprocessing, but is otherwise the same as IF GAN. These three IF GAN variants were tested against another GAN, WaveGAN [22], which operates in the phase domain on raw audio without generating spectrograms, and an autoregressive model, WaveNet, which is not a GAN [27].

The GANSynth team considered in their analysis five different metrics to evaluate the quality of produced sounds. The first metric was qualitative and based on human evaluation scores. Amazon's Mechanical Turk service was utilized where human listeners rated the quality of sounds that were presented to them for listening. The second method utilized was number of statistically different bins (NDB), where generated sounds were clustered into 50 cells and the number of cells where the number of samples in a cell is significantly different from the same cells generated from the training data is reported. The third metric considered was Inception score where the generated samples are evaluated via a neural network trained for classification on these sounds. The output of the neural network classifier is a model estimate of conditional distribution which is compared via Kullback-Leibler divergence to data distribution to evaluate the distance between the two. These distances are averaged over all examples used in the testing set. The fourth metric used in the study was quality of pitch which was evaluated via Pitch Accuracy and Pitch Entropy which measured accuracy and entropy by a pitch classifier. Finally, the GANSynth team examined Fréchet Inception Distance (FID) which calculates the Fréchet distance from the features identified in samples by an Inception classifier.

After analysis of the generated samples, the authors concluded that the IF method produced sounds that were most likely to be the highest rated relative to the other methods used in the study. IF methods outperformed the other methods in several hyperparameter runs where the authors varied the frequency resolution of the input features over all five metrics. Furthermore, the team concluded that IF methods generate high-quality sounds many times faster than autoregressive models.

While the GANSynth team achieved impressive results, the introduced method is computationally expensive given that their GANs were trained on an NVIDIA v100 GPU for four and a half days. The NVIDIA v100 GPU graphics cards are currently the best performing GPUs for deep learning [28]. Compared to a method like WaveGAN [22], which operates on raw audio, IF methods require substantial preprocessing of the data. STFTs, phase unwrapping, and scale shifting are all computationally expensive operations that must be performed on each sample in the input pipeline. A raw audio GAN implementation has to do none of these operations.

2.3 Antialiasing

2.3.1 Making Convolutional Networks Shift Invariant Again

In [29], author Richard Zhang outlines the issue of convolutional networks not being shift invariant. A system is shift invariant if a small shift perturbation in a systems causes only a small corresponding shift in the systems' output. Non-shift invariant systems exhibit large changes in outputs to small shifts in the systems' input. Zhang proposes a solution to fix that issue by incorporating antialiasing. Shift invariance is related to the linear shift invariant property of linear systems where a shift in the input produces linear shift in the output. For neural networks a qualitative description would be that the input to the neural network convolution layer should not cause non-linear changes to layer output. Shift variance is especially strong in max pooling layers and convolutional layers with strides larger than two and causes aliasing. Aliasing is related to sampling, specifically to an incorrect downsampling process where high-frequency components appear in lower frequency bands after downsampling. Given that the goal of having a convolutional architecture is to progressively reduce feature space, using a convolution with stride one then a pooling layer for dimensionality reduction (downsampling) would achieve this goal without a stride larger than two, but the aliasing problem would still persist. Although using average pooling instead of max pooling to downsample can help minimize the effects of aliasing, max pooling performs better than average pooling in machine learning systems which has led to its widespread adoption in convolutional networks.

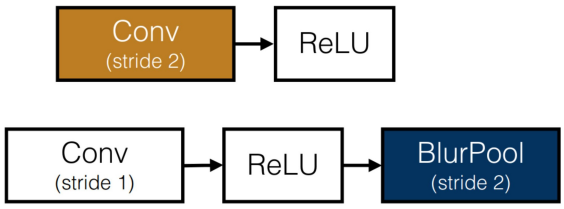

Figure 2.3. The top row shows a conventional max pooling convolution layer in a network. The bottom row shows how to apply blurring filters to achieve the same characteristics of a max pooling convolution layer while improving the layer's shift invariance [29]. Notice the convolution is no longer strided; downsampling is performed by the new blurring pool layer.

In signal processing, aliasing during downsampling is prevented by applying low-pass filters.

In [29], Zhang hypothesizes applying a blurring filter after an activation function may make modern convolutional layers more shift invariant while maintaining the task performance of max pooling. A diagram of a proposed architecture can be seen in Figure 2.3.

To see if adding a low-pass blurring filter would improve convolutional network performance three sets of experiments were performed. In each experiment, blurring pool layers were added to existing convolutional network architectures. The three filters tested can be seen in Table 2.3.

Table 2.3. Blurring pool layer filters from [29].

Filter Name	**Filter Value**
Rectangle-2	$[1, 1]$
Triangle-3	$[1, 2, 1]$
Binomial-5	$[1, 4, 6, 4, 1]$

The first experiment was a "dissection" of the VGG-13bn architecture. The purpose of this experiment was to see how shifting an image caused the values inside of a neural network to change. Two variants of the VGG-13bn architecture were dissected. The first variant was unmodified while the second variant was modified to include a blurring pool layer after each convolution and in lieu of each max pool layer. Shift-equivariance maps were then created of the two VGG-13bn variants by running unaltered and shifted examples from the CIFAR-10 dataset through the variants. The shift-equivariance maps showed that while the blurring pool layers made the modified variant much more, though not completely, shift-invariant.

The second blurring pool layer test focused on classifiers. Image classification was performed on eight popular image classifiers both unmodified and modified with the introduction of a blurring pool layer after each stride two convolution layer and in place of each max pool layer. Two experiments were performed for this test. The first focused on consistency through image shifts. Each classifier predicted the label of images from the ImageNet dataset that have been and have not been shifted. The classifiers' outputs were measured for accuracy and consistency, where accuracy is the percentage of time that the classifier predicted the correct label and consistency is the percentage of time the classifier assigned the same label to both a shifted and an unshifted version of the same image. In all cases,

the classifiers with the blurring pool layers had higher consistency and accuracy.

The second experiment performed during the classifier test measured stability to perturbations and robustness to corruption with and without blurring filters. This experiment used the ImageNet-P dataset which has short videos consisting of a single image from ImageNet with the addition of small perturbations like zooming, simulated weather, rotations, or random noise. To provide additional data, the ImageNet-C dataset was used which is similar to ImageNet-P but has additional perturbations added. Both blur pool layer modified and unmodified classifiers were run on the ImageNet-C and ImageNet-P datasets and the flip rate was measured. The flip rate is how often the classification changed on average in consecutive frames. The blur pool modified classifiers had a lower flip rate than the unmodified classifiers indicating that the blur pool layers assisted not only in making the classifiers more shift invariant but more robust in general.

Applying the blurring pool layer to conditional image generation was the last test. Images were generated using U-net. As in the past two tests, two versions of U-net were used, an unmodified version and a version where strided convolution layers are replaced with unstrided convolution layers followed by a blur pool layer. U-net contains upsampling as well as downsampling convolutional layers. A blur pool layer was inserted after each upsample layer. To measure the blur pool layer's impact on shift invariance, images were generated with a shifted and unshifted input and peak signal-to-noise ratios were compared between the outputs. Using larger filters in the blur pool layer led to images with higher peak signal-to-noise ratios, meaning the images were more shift invariant. A downside of the larger filters was the total variation of the generated images went down meaning the blur pool filters were removing more high-frequency detail. Qualitatively, the author assessed that the triangle-3 filter was the best compromise between shift invariance and high-frequency detail.

Ultimately [29] concluded that introducing antialiasing to convolutional networks has the ability to improve their performance. While [29] suggests that the convolution layers in GANs make them a strong candidate for the introduction of antialiasing techniques, no experiments were performed using a GAN.

CHAPTER 3:
Datasets

3.1 Speech Commands 0 through 9

The *Speech Commands 0 through 9* (SC09) dataset is a subset of the *Speech Commands* [25] dataset created by Donahue, McAuley, and Puckette [22]. While *Speech Commands* contains samples of twenty different words, SC09 only contains samples the spoken digits zero through nine. The samples in SC09 were recorded in an "uncontrolled" environment because the recordings were made by anonymous volunteers on their personal devices through a website for [25]. Both the original *Speech Commands* and SC09 datasets are open source datasets covered by the Creative Commons BY 4.0 license [30]. Samples in SC09 have different levels of background noise, are made by different speakers, and have sounds aligned differently inside of the recordings making a diverse and realistic dataset.

The motivations for using SC09 are very similar to the authors of [22]. Each sample in SC09 consists of a one second long recording of a speaker saying a digit between zero and nine sampled at 16 kHz in a single channel. Each sample therefore consists of $16,000$ data points. The size of the recordings is important because the larger the sample size, the more trainable weights there are in a GAN. A sample length of $16,384$ datapoints was chosen because the length already requires approximately 40 million trainable weights and produces a sample length of about one second at a 16 kHz sampling rate. A $16,384$ sample length also matches closely to the length of the SC09 sample lengths. The difference of 384 samples are filled by zeroes in our data pipeline implementation.

Another motivation to use the SC09 dataset is that it is easy for a human listener to tell if a GAN is generating a high-quality sound or not. This provides an advantage over other datasets because people frequently hear and are familiar with spoken digits. Identifying artifacts audibly is therefore trivial because artifacts make generated sounds mechanical. This is harder to do in the underwater acoustic dataspace because certain sounds, like ship propulsion, already sound machinelike.

The final motivation for choosing that SC09 lends itself well to benchmarking. SC09

consists of ten classes, one for each digit, reminiscent of the MNIST dataset. Each of these classes has between $2,352$ and $2,377$ samples such that the occurrence of each digit is uniformly distributed making a balanced dataset. The uniform distribution makes sounds generated by the SC09 dataset well suited for an Inception score (IS) metric because the IS for each digit should be approximately the same. Poorly performing GANs can quickly be identified by Inception scores that are either low or are widely distributed per digit. SC09 has also been used in other audio-generation testing in literature [22] letting our architecture be directly benchmarked against others.

3.2 Pure Tone Sinusoidal Waves

The use of a second dataset is necessary for measuring the amount of distortion introduced into a signal by GAN architectures. The idea behind this dataset is that a GAN trained on pure tone sinusoids should generate pure tone sinusoids. Parasitic harmonics present in generated sounds would be the result of aliasing or nonlinearities in the GAN. Since SC09 samples are composed of many different frequencies and harmonics determining exactly which frequencies are correctly part of a sound and which are artifacts is an intractable problem. By having a pure tone harmonic dataset this problem becomes much easier. Pure tone sinusoids only have one frequency. All other present frequencies are parasitic harmonics, making the problem of identifying which frequency is part of a sound trivial.

The pure tone dataset consists of $50,000$ .wav files sampled at 16 kHz generated by a Python program. Tones were generated with one of twelve fundamental frequencies ranging from 200 Hz to 8 kHz, 8 kHz being the upper limit of sound that can be properly represented in 16 kHz sampling rate according to the Nyquist Sampling Theorem. Nine of the fundamental frequencies are in the lower half of the frequency band compared to the three in the upper half of the frequency band because the human ear is more sensitive to changes in frequency at lower ranges [31]. The occurrence of tones generated by each fundamental frequency is uniformly distributed across the dataset. To improve the diversity of the dataset and to prevent GAN architectures from memorizing twelve distinct samples, a random phase and amplitude was assigned to each sample in the pure tone dataset. The analytical description of simulated signals is given with the equation

$$\psi(t) = A \sin(\omega t + \phi),$$

where $\psi(t)$ is the generated tone, A is a randomly assigned amplitude from 0 to 1, ω is the fundamental frequency in radians ($\omega = 2\pi f_0$), t is time step, and ϕ is a randomly assigned phase from 0 to 2π. The generated tones were encoded into .wav format using the Tensorflow 2.0 audio module wave encoding function [32].

3.3 Underwater Acoustics

The final dataset is a collection of underwater samples from the Oceanography Department of the Naval Postgraduate School. This acoustic data was collected by the US Navy at Thirty Mile Bank about 25 nm west of Point Loma, California, with a high-frequency acoustic recording package at a depth of approximately 380 fathoms [33].

This dataset was used for training on high-quality ship noises. The dataset contains hours of recordings of the full underwater soundscape. To extract time segments that contained ship noise, the underwater recordings were correlated with Automatic Identification System (AIS) data and time segments with a ship within $4,300$ yards of the acoustic recording package were saved and broken into 30-second time segments by [34]. All the samples in this dataset have a 4 kHz sampling rate. These samples were then broken up into segments $16,384$ samples long so that they could be used to train GANs.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
TangGAN

4.1 TangGAN Architecture

TangGAN[1] is a Generative Adversarial Network (GAN) for synthesizing audio. TangGAN was developed as an improvement to WaveGAN by incorporating anti-aliasing decimation filters in the discriminator and traditional interpolation in the generator. These additions are designed to remove aliasing artifacts introduced by improper sampling in modern convolutional layers with strides larger than one, as discussed extensively in [29].

Like all GANs, TangGAN consists of a discriminator and a generator that are trained by a loss function. TangGAN is trained via the Wasserstein GAN Gradient Penalty (WGAN-GP) method [24]. WaveGAN and TangGAN are both based on DCGAN, an architecture that has its own unique training method [23]. While WaveGAN preserves most of the DCGAN architecture it replaced the DCGAN loss function with the WGAN-GP loss because its creators assessed this change generated better sounds [22]. An early prototype of TangGAN was trained using the DCGAN method and confirmed the claim in [22] that more intelligible sounds were created by adopting the WGAN-GP method.

4.2 TangGAN Discriminator

The discriminator makes an estimation on whether a sample is from the real or generated dataset. This is accomplished by taking in a sample of length 16, 384 and through compression of information over five convolutional layers to produce an estimate of the sample's dataset origin, real or fake via a binary classification. The purpose of the convolutional layers is therefore two-fold. The first is to learn a data representation from the inputs and the second is to downsample, or reduce the input feature space, so that a single determination of the input's origin can be made. WaveGAN accomplishes this through the five one-dimensional convolutional layers, but as demonstrated in [29] this results in improper

[1]TangGAN is named after the USS *Tang* (SS 306), the most successful American submarine of World War II, commanded by CDR Richard O'Kane.

sampling and introduces checkerboard style artifacts. As discussed in Chapter 2, the authors of WaveGAN recognized this problem but failed to address it and instead managed its consequences through the phase shuffle layer [22].

TangGAN addresses the problem of improper sampling through the introduction of anti-aliasing filters. These anti-aliasing filters remove the task of compressing the input data space from the convolutional layers and leverage anti-aliasing decimation layers instead. The difference between downsampling and decimation is that downsampling refers to discarding samples without a low-pass filter to address aliasing while decimation uses a low-pass filter prior to compressing input by discarding samples [35]. TangGAN constructs a decimation layer by simultaneously sampling inputs and smoothing with an anti-aliasing filter. A decimation layer is stacked after each convolutional layer in the discriminator. The stride of each convolution is set to one so that the convolutional layers do not perform any downsampling. Figure 4.1 shows how the convolutional and decimation layers are combined in TangGAN to prevent aliasing.

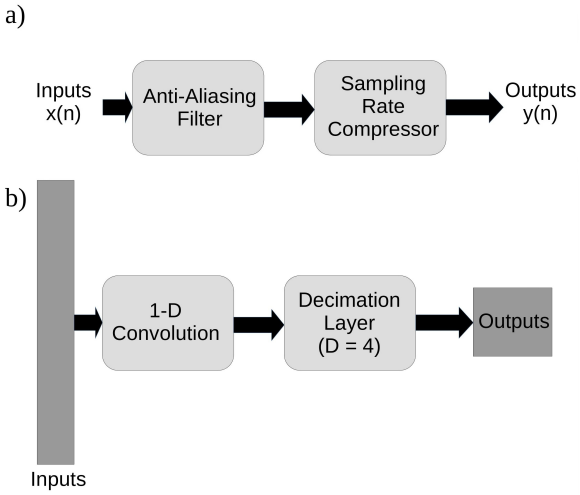

Figure 4.1. A typical decimation layer in TangGAN. Diagram a) shows how decimation layers combine anti-aliasing filtering with sampling rate compression to perform proper decimation. Diagram b) shows how the decimation layer is incorporated inside of TangGAN. Inputs are fed to a single-stride, one-dimensional convolution which learns features. The convolution's output is then compressed by the decimation layer to avoid aliasing.

In TangGAN, the size of the decimation layer output must always be the size of the input

divided by the decimation factor regardless of filter size. The inputs are compressed by applying the decimation filter through a convolution with a stride of one across the input feature space. In cases where the input filter cannot be strided across the input feature space x times where $x = \frac{\text{input feature space}}{\text{decimation factor}}$, the output layer will be decimated more than the decimation factor. To prevent this from happening, TangGAN decimation layers pad the layer's input via the method suggested in [29]. To pad the input, $\lfloor \frac{\text{filter length}}{2} \rfloor$ values are added prior to the beginning of the input and $\lceil \frac{\text{filter length}}{2} \rceil$ are added after the end of the input. Any numerical value can added by the padding to ensure that the filter can be strided x times across the input feature space. By default, Tensorflow applies zero-padding; however, this is known to produce artifacts at the boundaries because the features are combined with the padded zeroes by the decimation filter. A better choice is to use reflection padding which fills padded values with a reflected version of the input. An example of why TangGAN decimation layers requires padding and how it is implemented can be seen in Figure 4.2.

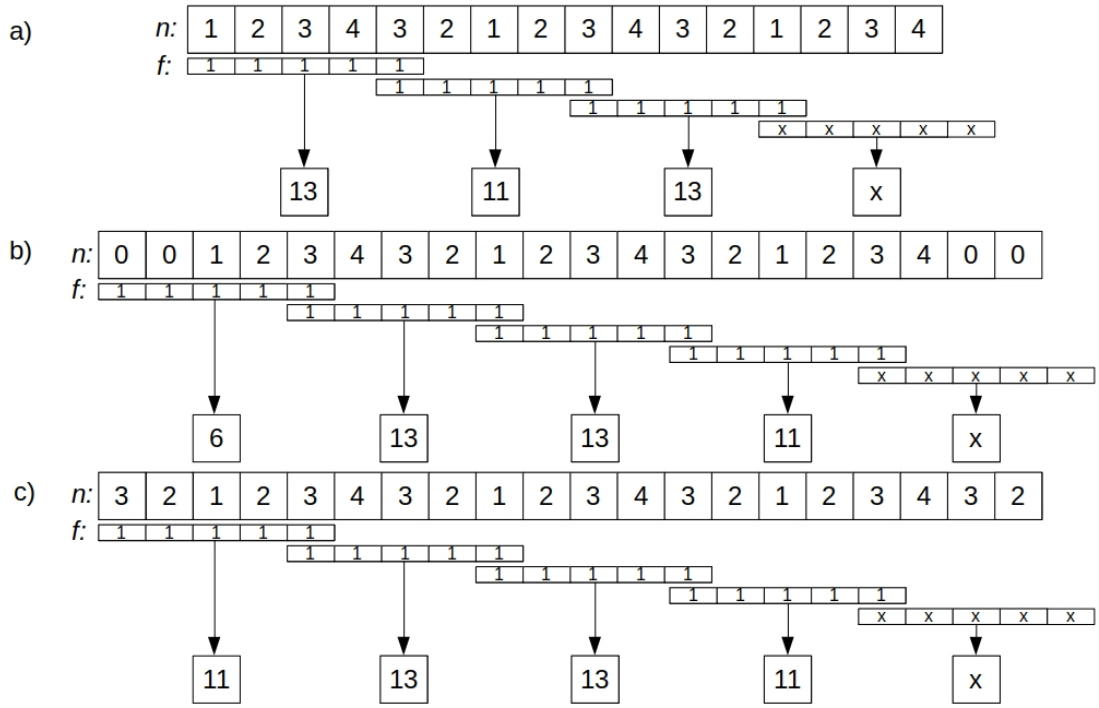

Figure 4.2. Demonstration of a decimation layer decimating input, n (length 16), by a factor of four by striding filter, f (length of 5). In a), n is not padded. Since the last stride of f does not have enough values in n to operate, it is invalid and does not produce a value for the output. The output of the decimation filter is only three values long meaning the input has been decimated by more than the decimation factor. In b), n is padded on the left by $\lfloor \frac{\text{filter length}}{2} \rfloor$ zeroes on the right by $\lceil \frac{\text{filter length}}{2} \rceil$. The output has been decimated by the decimation factor, but features at the beginning have been suppressed creating an edge artifact because the beginning values are combined with the padding zeroes by the filter. In c) n is padded by the same amount as in b) except this time n is padded by its own reflection. This results in an output that preserves the output size decimation factor relation without an edge artifact.

TangGAN implements decimation layers by creating a custom Tensorflow layer. The layer pads the inputs so that the output size is determined by the decimation factor, loads

an anti-aliasing filter, then performs a depthwise convolution on the padded input with a stride equal to the decimation factor using the Tensorflow neural network module's depthwise convolution implementation. The depthwise convolution in the decimation layer therefore downsamples the inputs and then applies an anti-aliasing filter effectively applying decimation.

The entire architecture for the TangGAN discriminator can be seen in Table 4.1. Each convolution layer is followed by a leaky rectified linear unit (LeakyReLU) function which serve as activation functions. This architecture is different than WaveGAN's in that each convolutional layer has a stride of one and is followed by a decimation layer. The final layer in the discriminator uses a linear activation function. It was experimentally determined that a non-linear activation function after the final layer in the discriminator caused TangGAN to rapidly diverge.

Table 4.1. The TangGAN discriminator architecture. D = decimation factor, s = stride, b = batch size.

Layer	**Arguments**	**Output Dimension**
Input	—	$(b, 16384, 1)$
1D Convolution	$s = 1$	$(b, 16384, 64)$
LeakyReLU	—	$(b, 16384, 64)$
Decimation	$D = 4$	$(b, 4096, 64)$
1D Convolution	$s = 1$	$(b, 4096, 128)$
LeakyReLU	—	$(b, 4096, 128)$
Decimation	$D = 4$	$(b, 1024, 128)$
1D Convolution $s = 1$	—	$(b, 1024, 256)$
LeakyReLU	—	$(b, 1024, 256)$
Decimation	$D = 4$	$(b, 256, 256)$
1D Convolution	$s = 1$	$(b, 256, 512)$
LeakyReLU	—	$(b, 256, 512)$
Decimation	$D = 4$	$(b, 64, 512)$
1D Convolution	$s = 1$	$(b, 64, 1024)$
LeakyReLU	—	$(b, 64, 1024)$
Decimation	$D = 4$	$(b, 16, 1024)$
Flatten	—	$(b, 16384)$
Dense	—	$(b, 1)$

4.2.1 Decimation Filter Design

The choice of filter inside of the decimation layers is consequential. A filter needs to attenuate high-frequency features to prevent aliasing while keeping low-frequency to preserve features. Conversely, a filter that attenuates too much low-frequency data risks obliterating the features necessary for the classification of sounds, while a filter that fails to attenuate enough of the high-frequency data will not prevent aliasing. A theoretical ideal filter would therefore pass all the low-frequency data so that no input features are degraded and attenuate all the high-frequency data to remove aliasing in the decimation layer's output. In practice, however, this is not possible; a trade-off between low-frequency passing and high-frequency attenuation is required.

Mathematically, the operation of the decimation filter can be explained via the following [36]. An input sequence $x(n)$ is passed through a low-pass filter, $H_d(\omega)$ that satisfies the conditions of

$$H_D(\omega) = \begin{cases} 1 & |\omega| \leq \pi/D \\ 0 & \text{otherwise} \end{cases}.$$

The ideal filter therefore leaves the frequencies of signal $X(\omega)$ in the passband $0 \leq \omega \leq \pi/D$ unaltered and completely removes frequencies above the passband in the range $\pi/D < \omega < \pi$.

While this thesis did not undertake a study of designing the best possible decimation filter, three different filters were analyzed and two selected for implementation in TangGAN decimation layers. A comparison of the three decimation filters can be seen in Figure 4.3. The decimation layers in TangGAN all have a decimation factor of four meaning the desired cutoff frequency for the decimation filter passband is 0.25 of the normalized frequency. The cutoff frequency is inversely proportional to the decimation factor. An ideal filter would therefore pass all of the signal below 0.25 normalized frequency and reject all of the signal above.

The first filter analyzed was a length seven binomial filter used in [29]. A binomial filter of length n consists of the first row of numbers from Pascal's triangle containing n numbers. While the $n = 7$ binomial filter attenuates frequencies above the cutoff frequency by -17 dB or more, it significantly attenuates frequencies in the pass band. The $n = 7$ binomial

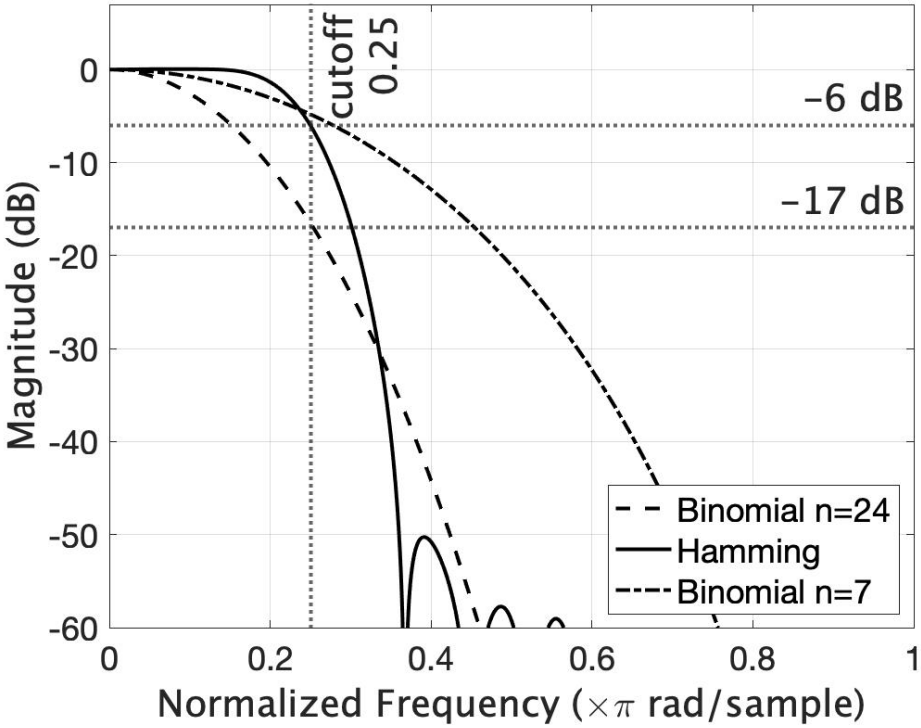

Figure 4.3. Decimation filters compared by attenuation magnitude versus normalized frequency. The ideal decimation filter would have 0 dB attenuation until the cutoff frequency and $-\infty$ dB attenuation after. The $n = 7$ binomial filter provides good attenuation outside of passband, but rejects too much of the signal below the cutoff frequency. The $n = 25$ binomial filter attenuates less in the passband, but does not attenuate effectively above the cutoff frequency. The custom designed Hamming FIR filter minimizes attenuation in the passband while maximizing attenuation outside.

filter was not incorporated in TangGAN because of it attenuates too much of the signal in the pass band and could hamper the discriminator's learning through the unacceptable input degradation over five decimation layers.

The second filter analyzed is an $n = 25$ binomial filter. The $n = 25$ filter has a maximum attenuation of -6 dB in the passband meaning it does not suffer from the same passband attenuation the $n = 7$ binomial filter does. The $n = 25$ binomial filter trades performance in the passband for decreased rejection above the cutoff frequency. The $n = 25$ does not match the -17 dB attenuation of the $n = 7$ binomial filter until more than twice the cutoff

frequency. This decimation filter was experimentally tested in a TangGAN discriminator.

The final filter analyzed was a custom filter created to approximate the ideal filter as much as possible by passing signals below the cutoff frequency and rejecting signals above. This custom filter is a finite impulse response (FIR) filter designed using a Hamming window-based approach for a decimation factor of four. The resulting custom filter is a 31^{st} order filter and will be referred to as the Hamming FIR filter. The Hamming FIR filter passes more of the signal in the passband than either binomial filter and rapidly attenuates signal outside of the passband. The attenuation in the passband is nearly flat until the cutoff frequency of -6 dB. The -17 dB attenuation of the $n = 7$ binomial filter rejection band is achieved at 1.2 times the cutoff frequency. The Hamming FIR filter is the best quality decimation filter analyzed and was used in TangGAN decimation layers.

4.3 TangGAN Generator

The generator is responsible for growing a size 100 latent vector into a sound sample of length $16,384$. WaveGAN accomplishes this through upsampling via five transpose convolutional layers. Upsampling is the process of inserting zeroes between datapoints without addressing aliasing [35]. The filter inside of the transpose convolution is therefore responsible for ensuring that audio features are generated and learning a filter response that avoids aliasing. These dual responsibilities can be conflicting during training. Interpolation, in the signal processing sense, is following upsampling with a low-pass filter. Since this filter is fixed, the responsibility of learning an anti-aliasing filter response can be removed from the convolutional layers, allowing them to focus on learning audio features. TangGAN allows convolutional layers to focus on learning audio features through the introduction of interpolation layers in the generator. A diagram showing the coupling of interpolation and convolutional layers in TangGAN can be seen in Figure 4.4.

The TangGAN architecture differs from the WaveGAN architecture in that transpose convolutional layers are not used. To grow a latent vector into a sound sample, inputs are grown by an interpolation layer. The interpolation layer is then followed by a stride one convolutional layer whose sole purpose is to learn how to produce audio features. Since the interpolation layer manages all of the dimension changes for the generator, a single stride convolutional and a transpose convolutional layer both perform the same operation;

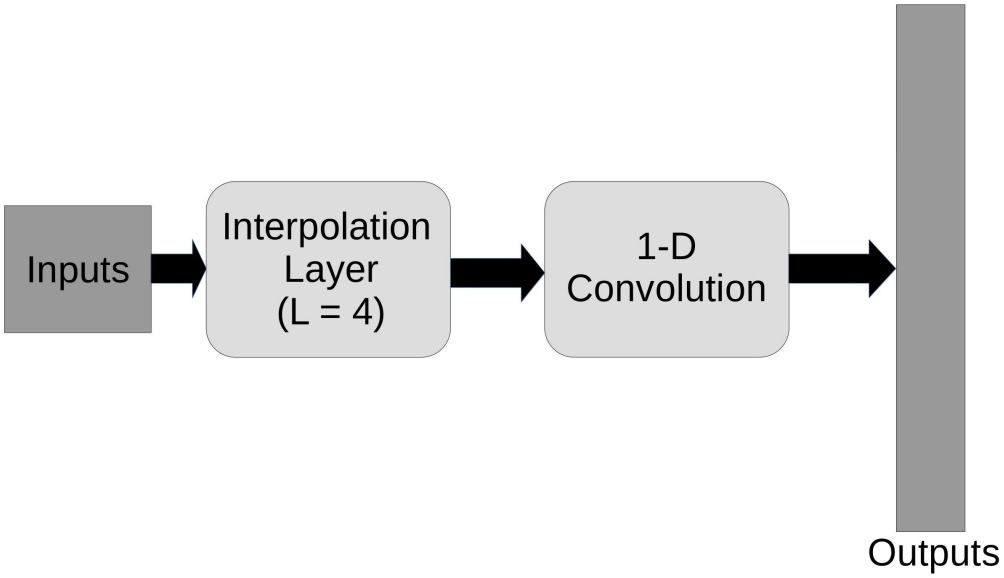

Figure 4.4. A typical interpolation layer in TangGAN. Layers are expanded by a factor of four through a traditional interpolation layer. The expanded inputs are then given to a single stride convolutional layer which only has to learn how to produce audio features.

but, the convolutional layer does so more efficiently in Tensorflow. Furthermore, since Tensorflow has a one-dimensional convolutional layer but only higher dimension transpose convolutional layers, the need to add and strip dimensions on the input is avoided reducing computational complexity.

The complete TangGAN generator architecture is presented in Table 4.2. A latent vector of size 100 is turned into a sample of size 16 with $1,024$ different channels by a dense and reshaping layer. This size 16 sample is then grown into a size $16,384$ sample over four stacks consisting of an interpolation, a one-dimensional convolution, and a rectified linear unit (ReLU) layers. A fifth and final stack replaces the ReLU activation function with a hyperbolic tangent function. Each interpolation layer grows the sample by a factor of four.

Table 4.2. The complete TangGAN generator architecture. L = growth factor, s = stride, b = batch size.

Layer	Arguments	Output Dimension
Input	—	$(b, 100)$
Dense	—	$(b, 16384)$
Reshape	—	$(b, 16, 1024)$
ReLU	—	$(b, 16, 1024)$
Interpolation	$L = 4$	$(b, 64, 1024)$
1D Convolution	$s = 1$	$(b, 64, 512)$
ReLU	—	$(b, 64, 512)$
Interpolation	$L = 4$	$(b, 256, 512)$
1D Convolution	$s = 1$	$(b, 256, 256)$
ReLU	—	$(b, 256, 256)$
Interpolation	$L = 4$	$(b, 1024, 256)$
1D Convolution	$s = 1$	$(b, 1024, 128)$
ReLU	—	$(b, 1024, 128)$
Interpolation	$L = 4$	$(b, 4096, 128)$
1D Convolution	$s = 1$	$(b, 4096, 64)$
ReLU	—	$(b, 4096, 64)$
Interpolation	$L = 4$	$(b, 16384, 64)$
1D Convolution	$s = 1$	$(b, 16384, 1)$
tanh	—	$(b, 16384, 1)$

4.3.1 Interpolation Implementation

The selection of an interpolation method requires the balancing of two competing demands, the quality of interpolation and the cost of interpolation. Six different interpolation methods were analyzed for their suitability in a TangGAN interpolation layer. The cost of the interpolation methods is presented in Table 4.3 and a demonstration of the effectiveness of interpolation methods can be seen in Figure 4.5.

In signal processing, the standard method of interpolation is the insertion of zeroes between

Table 4.3. A table of computational cost for interpolation methods. The cost reported is the number of seconds it took to train one mini-batch of 64 SC09 samples inside of a TangGAN interpolation layer. The standard signal processing interpolation method and bicubic interpolation method were not used in any further experiments.

Interpolation Method	Cost in seconds	Feasible
Standard Signal Processing (scatter)	1200	No
Zero Order Hold and Smoothing Filter	6	Yes
Bilinear	4	Yes
Mitchell Cubic	20	Yes
Bicubic	30	No
Lanczos-5	34	Yes

data points followed by a low-pass filter. Three implementations of the signal processing method were attempted in Tensorflow. The first implementation naïvely used Tensorflow to loop through a signal and insert zeroes between data points one at a time. This method was so computationally expensive it was abandoned during the testing phase. The second method made use of the Tensorflow neural network module transpose convolutional layer. A transpose convolutional layer with the right kernel can be made to upsample at low computational cost. The problem with this method is that no depthwise transpose convolutional layers are implemented in Tensorflow meaning that when more than one channel is present in the input, all the channels become mixed together and the sample is not properly upsampled. Channel mixing forced this implementation of the standard signal processing method to be abandoned. The final implementation created a new tensor of zeroes in the correct upsampled dimensions and used the Tensorflow scatter_nd method to place the sample data in the correct locations in the upsampled tensor. A low-pass filter was then applied. While this method was faster than the naïve implementation and performed the upsampling correctly, it was too computationally costly to use in TangGAN.

The second interpolation method analyzed was a zero-order hold followed by a low-pass filter. The zero-order hold was implemented using the efficient Tensorflow repeat method which makes it one of the fastest interpolation methods. The low-pass filter after the zero-order hold is essential to ensure that this method interpolates smoothly instead of producing a square wave interpolation which could cause the introduction of artifacts. The Hamming

FIR filter used in the decimation layer was used as the low-pass filter for the zero-order hold as well.

The remaining four interpolation methods, bilinear, bicubic, Mitchell cubic, and Lanczos-5, all implemented interpolation through the Tensorflow image module. As the Tensorflow image module is designed to operate on two-dimensional images instead of one-dimensional audio samples, the audio samples required an extra dimension to be added prior to interpolation and which was stripped immediately after the interpolation. The quality of these interpolation methods is inversely proportional to their cost, the bilinear method having the worst quality but lowest cost and the Lanczos-5 method having the best quality but highest cost.

Ultimately, after analysis, the standard signal processing method was determined to be too expensive to incorporate into a TangGAN interpolation layer for later experiments. The bicubic interpolation method was also abandoned because it was nearly as expensive as the Lanczos-5 method but had a lower quality interpolation. TangGAN interpolation layers deemed feasible for later experiments were zero-order hold with smoothing filter, bilinear, Mitchell Cubic, and Lanczos-5 interpolation.

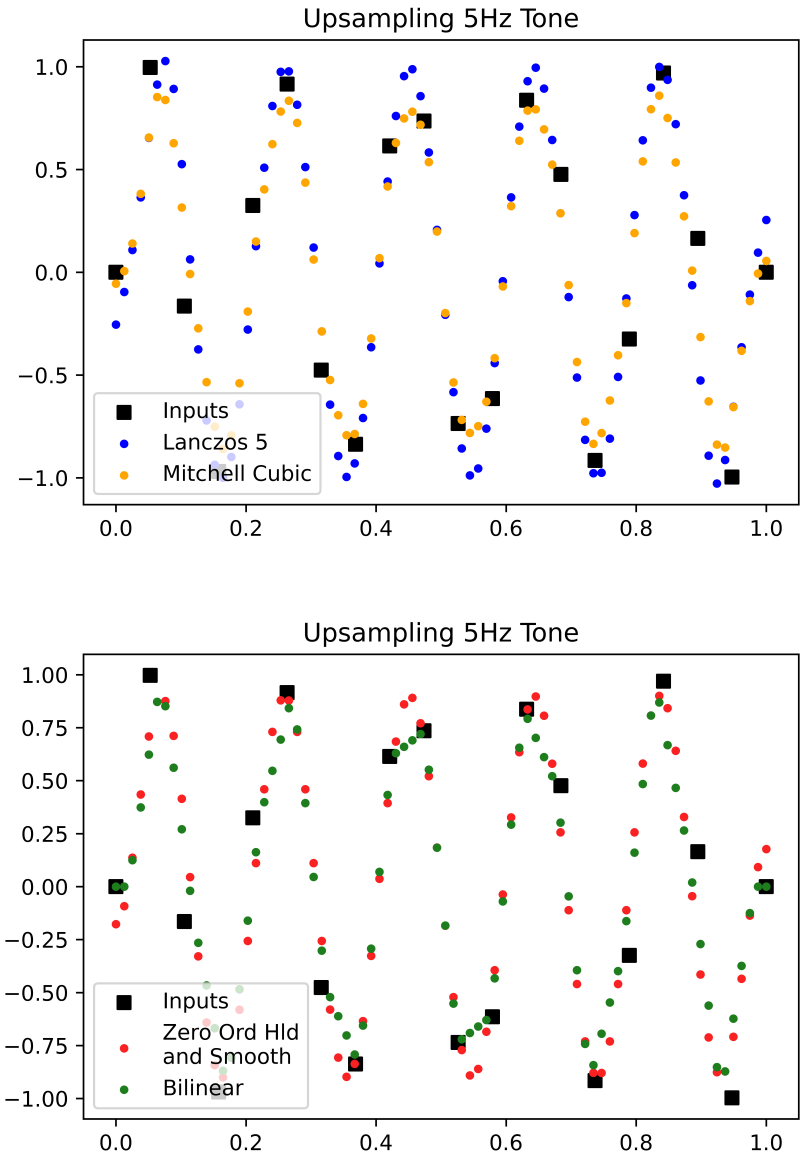

Figure 4.5. Example of methods interpolating a 5 Hz sinusoid. The methods in the upper graph provide higher quality interpolations but are more computationally expensive. The higher quality interpolations produce peak data points closer to -1 and 1 where a 5 Hz sinusoid's peak points would be. Higher quality interpolations also provide a smoother, more continuous trail of datapoints throughout the graphs.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Experimental Design and Evaluation Methodology

Four experiments were performed on TangGAN to evaluate performance. The first experiment, Architecture Exploration, served to determine which architectures would be most successful to incorporate into TangGAN. The second experiment, SC09 Generation, used the most promising TangGAN architectures to generate SC09 digits to benchmark TangGAN to the state-of-the-art WaveGAN. The third experiment, Pure Tone Generation, was conducted to establish a quantitative metric of artifacts generated with sounds. The fourth and final experiment, generated underwater acoustic samples, is a proof-of-concept that TangGAN can be used for data generation in this DoD-relevant domain.

5.1 Architecture Exploration

5.1.1 Experimental Design

This experiment explores the viability of TangGAN architectures. TangGAN is configurable so that different decimation filters and interpolation methods can be chosen. An experiment to rapidly develop and identify the best TangGAN prototype architectures without fully training them was necessary because even the simple WaveGAN took days to fully train. More complex TangGAN architectures would likely take even longer. This experiment focused on rapid prototyping and developing an understanding whether the integration of decimation and interpolation layers could generate sounds identifiable as a digit with a limited amount of training.

All the training for this experiment was conducted on a single NVIDIA Titan RTX GPU on the infrastructure provided by the High Performance Computing Center at the Naval Postgraduate School. Each prototype trained for a fixed number of hours on the SC09 dataset. The WaveGAN architecture and TangGAN architectures with only a modified discriminator were trained for 16 hours; this was determined experimentally to be enough training time for a GAN to generate samples that were identifiable as a digit. The TangGAN architectures that incorporated interpolation layers in the generator were trained for 32 hours

Table 5.1. TangGAN Architecture Prototypes

Prototype Name	**Discriminator Architecture**	**Generator Architecture**	**Training Hours**
WaveGAN	WaveGAN with Phase Shuffle	WaveGAN	16
Binomial Filter Decimation	Binomial Filter with Zero Pad	WaveGAN	16
Binomial and Mirror Pad Decimation	Binomial Filter with Mirror Pad	WaveGAN	16
Hamming FIR Decimation	Hamming FIR Decimation with Mirror Pad	WaveGAN	16
Smoothing Filter Interpolation	Hamming FIR Decimation with Mirror Pad	Zero Order Hold and Smoothing Filter	32
Bilinear Interpolation	Hamming FIR Decimation with Mirror Pad	Bilinear Interpolation	32
Mitchell-Cubic Interpolation	Hamming FIR Decimation with Mirror Pad	Mitchell-Cubic Interpolation	32
Lanczos Interpolation	Hamming FIR Decimation with Mirror Pad	Lanczos-5 Interpolation	32

each. The increase in training time was necessary due to the increased complexity of the generator.

Seven TangGAN architectures were tested. Three architectures incorporated decimation layers in the discriminator. An additional four architectures incorporated decimation layers in the discriminator and interpolation layers in the generator. The addition of decimation layers in the discriminator tests the hypothesis that the discriminator introduces aliasing artifacts which can be reduced through decimation layers. The addition of interpolation layers tests the hypothesis that the generator can better learn features if the task of low-pass filtering is handled by traditional interpolation layers. To establish a baseline the best published WaveGAN model architecture was trained [22] on the same dataset. A list of all tested architectures is presented in Table 5.1.

The first three prototypes were designed to reduce aliasing inside of the discriminator architecture. Aliasing is characterized by a checkerboard pattern. These prototypes introduce decimation layers after each convolutional layer in discriminator architectures. Besides the introduction of the decimation layers, the discriminator architectures differed from the WaveGAN discriminator in that each convolution had a stride of one and given correct

treatment of aliasing there was no need for a phase shuffle layer as in original WaveGAN architecture. As a reminder, phase shuffle layer was used to address the consequences of aliasing. A more in-depth discussion on how the decimation layers were designed and incorporated in the discriminator can be found in Chapter 4.

The first of the decimation layer prototypes, the binomial filter decimation prototype, used an $n = 25$ binomial filter layer with zero padding in the decimation layers. This prototype architecture tested the viability of a binomial filter decimation layer. The second decimation layer prototype, binomial filter decimation with mirror padding, used the same $n = 25$ binomial filter but with mirror padding instead of zero padding to test if mirror padding reduces edge artifacts. The third and final decimation layer prototype is the Hamming Finite Impulse Response (FIR) decimation prototype. This prototype uses a 31^{st} order low-pass Hamming FIR filter with mirror padding in the decimation layer. The Hamming FIR filter was designed as a better low-pass filter for anti-aliasing compared to binomial design.

The next four TangGAN prototypes were designed to address aliasing in upsampling. This was achieved by incorporating interpolation layers in TangGAN generator architectures. These TangGAN prototypes all introduced interpolation layers before the convolutional layers. All four of these prototypes use the Hamming FIR decimation discriminator prototype to prevent aliasing in the discriminator. Besides the introduction of the interpolation layers, these prototypes' generators differ from the WaveGAN generator in that stride four transpose convolutional layers are replaced by stride one convolutional layers. A more detailed description of the prototype generator architecture can be found in Chapter 4.

The first TangGAN prototype to incorporate interpolation layers is the zero-order hold and smoothing filter prototype. This prototype uses a custom-created layer that most closely follows the traditional signal processing interpolation method. The other three interpolation prototypes, the bilinear, Mitchell Cubic, and Laczos-5 interpolation prototypes, performed interpolation using the Tensorflow image module implementations of those algorithms. All the interpolation prototypes were designed to test the viability of the interpolation algorithm used at generating viable output.

5.1.2 Evaluation Methodology

In this experiment, an architecture was deemed valid if it generated sounds that had an Inception score greater than two and generated sounds could be identified as digits by a human listener. Inception score (IS), first proposed in [37], establishes a method that provides a rough measure of sound quality and a quantitative metric that sometimes correlates with human perception [22]. WaveGAN results were also given in IS [22], allowing for a direct comparison between WaveGAN and TangGAN. To provide the most consistent comparison, the IS implementation from [22] is used. An in-depth discussion of IS can be found in Chapter 2, but as a reminder, IS is a comparison of probability distributions between generated and actual samples as measured by a classifier. In [22], sounds are used to create spectrograms and the spectrograms are classified by a pre-trained neural network classifier. The probability distributions of the real data and generated data from the classifier are used to calculate the IS using the following: $e^{\mathbb{E}_x D_{\mathrm{KL}}(P(y|x)||P(y))}$, where $\mathbb{E}_x$ is the expected probability distribution of x, D_{KL} is the Kullbeck-Leibler Divergence, and P are probability distribution functions. A higher IS indicates that an Inception style classifier is more confident in its classification and distribution of the generated samples. For a balanced ten-class dataset such as SC09, the maximum IS possible is ten. All IS were calculated using 50,000 generated samples from each architecture.

5.2 SC09 Generation

5.2.1 Experimental Design

Using the results from the architecture exploration experiment, the three most promising TangGAN architecture prototypes were trained on the SC09 dataset with the goal of generating the best sounds possible. The results published in [22] were obtained after training WaveGAN for four days on an NVIDIA P100 GPU. This approximates a week of training on the NVIDIA Quadro RTX 8000 GPUs this experiment was performed on. To produce the best sounds possible, the Hamming FIR Decimation filter, bilinear interpolation, and Lanczos-5 interpolation architectures were each trained for one week on the SC09 dataset. The WaveGAN architecture was trained for a week as a baseline for comparison. A more detailed description of these architectures is provided in Table 5.1 and the Architecture Exploration section of this chapter.

Along with performing WaveGAN and TangGAN architectures evaluations after a week of training, the WaveGAN, TangGAN Hamming FIR Decimation filter, and TangGAN bilinear interpolation architectures were all evaluated after training the same number of epochs regardless of training time. The TangGAN Lanczos-5 interpolation architecture was omitted from this comparison because the amount of training required would have taken over ten weeks while all other architectures could reach the same number of epochs in about two weeks.

5.2.2 Evaluation Methodology

Two metrics were used to measure the quality of produced sounds, Inception score (IS) and speech-to-reverberation modulation energy ratio (SRMR). IS is described in the previous section. While IS does well at giving a measure of similarity between features in generated and real sounds as determined by an audio classifier, it does not do well at measuring how artifacts affect the quality of sounds as heard by a human listener. Research in speech and language processing has developed several metrics that measure the human listener-perceived quality of sounds. In order to compare sounds generated by TangGAN, a non-intrusive, wide-band metric is required. Perceptual evaluation of speech quality (PESQ) [38] and other widely regarded metrics in speech and language processing are intrusive, meaning they measure the difference in quality between a distorted and non-distorted version of the same sound. Intrusive metrics must have access to both the input and output signal of the system being measured. Since TangGAN generates sounds that are uniquely created, distorted and non-distorted versions of the sound (signal) to measure do not exist. A non-intrusive metric not requiring a reference signal is therefore required. Additionally, evaluation of this experiment requires a wideband metric. Narrowband metrics such as P.563 [39] do not cover the full frequency spectrum of sounds generated by TangGAN and would require decimation of generated samples prior to evaluation, a process that could falsely remove artifacts and their effects in generated samples.

Speech-to-reverberation modulation energy ratio (SRMR) was originally developed to measure the effectiveness of dereverberation algorithms in increasing intelligibility of human speech [40]. Dereverberation is the process of removing the distortions in a sound due to sound propagation and reflections from boundaries. Speech enhancement algorithms often introduce artifacts and both intrusive and non-intrusive metrics are used to understand the

end-performance typically evaluating intelligibility of enhanced speech [41]. Traditionally, the effects of these artifacts have been evaluated through expensive human-listener evaluations by generating a mean opinion score (MOS) metric, in a process similar to the human-based evaluations on GAN generated samples in [22]. In order to avoid the necessity of human trials, in [40] a wideband, non-intrusive quality-of-sound metric is proposed and developed that correlates well with a MOS metric. The open-sourced implementation of SRMR by the authors of [40] available at [42] was used in this work. All reported SRMR scores are the average SRMR score from 50,000 generated samples. Higher SRMR scores indicate samples have less artifacts and should correlate to a higher MOS score as assigned by a human listener.

5.3 Pure Tone Generation

5.3.1 Experimental Design

This experiment provides a platform to quantitatively measure the occurrence of artifacts in generated sounds. In this experiment, WaveGAN, TangGAN Hamming FIR decimation, and TangGAN bilinear interpolation architectures were trained on the pure tone dataset. The Lanczos-5 interpolation architecture was omitted from this experiment as the SC09 generation experiment showed similar performance between the Lanczos-5 and bilinear interpolation architectures but the Lanczos-5 interpolation architecture took over five times as long to train per mini-batch. Each architecture in this experiment was allowed to train for 250 epochs on a single NVIDIA V100 GPU on an Amazon Web Service P3.2XLarge EC2 instance. Since the pure tone dataset consists of single frequency tones, any other tones present in the output would be due to artifacts.

5.3.2 Evaluation Methodology

Total harmonic distortion (THD) and signal-to-noise ratio (SNR) were used to measure the occurrence of artifacts in generated sounds. THD is a measure of the harmonic distortion present in a signal. Since a pure tone only has a fundamental frequency, any additional harmonic components present are deemed artifacts [43]. Artifacts caused by improper aliasing would cause harmonic distortions. A lower THD value indicates a sample with less harmonic distortion. SNR compares a signal to the amount of noise present in a sample.

Again, since a pure tone sample's signal is exclusively made up of the pure tone frequency, any artifacts present would contribute to noise in the sample and cause a lower SNR.

The MATLAB Signal Analyzer Toolbox implementation of THD and SNR was used for calculation [44]. THD is defined mathematically in [43] as a ratio of total harmonic content of the signal and the fundamental frequency. Specific implementation in MATLAB is given as $\text{THD} = 20\log_{10}\frac{\sqrt{\Sigma_{n=2}^{5} I_n^2}}{I_1}$ where I_1 is the fundamental frequency and I_n are harmonic components. The SNR is calculated in MATLAB by creating a modified periodogram with a Kaiser window with $\beta = 38$. Final reported THD and SNR values are the averages of $50,000$ generated samples for each architecture.

5.4 Underwater Acoustic Proof-of-Concept

5.4.1 Experimental Design

The final experiment in this work is a proof-of-concept that TangGAN can be used to generate sounds from underwater acoustic data. The base WaveGAN, TangGAN Hamming FIR decimation, and TangGAN binomial interpolation architectures were all trained for one week on two subsets of the underwater acoustics dataset [33]. One subset of the data featured sounds from small ships and the other from large ships. Small ships were defined as any ship with a tonnage less than or equal to $5,000$ and large ships were any ships with more tonnage than $5,000$. The dataset was split up in this manner so that TangGAN could be used to conditionally generate either small ship or large ship sounds. All training was performed on NVIDIA Quadro RTX 8000 GPUs.

5.4.2 Evaluation Methodology

As this experiment is a proof of concept on training on a unique dataset, generated sounds were evaluated qualitatively both audibly and via spectrograms. Ten spectrograms were created from randomly generated sounds. In spectrograms signal strength is represented as color intensity on time-frequency axes. The spectrograms were created using the Tensorflow signal module short-time Fourier transform (STFT) method. Each STFT window was 250 time steps long (16 milliseconds at a 4 kHz sampling rate) with an overlap of 200 time steps (13 milliseconds at a 4 kHz sampling rate). The edges of each sample was

padded with zeroes. After creation, the spectrograms of generated sounds were compared to spectrograms of actual acoustic data and evaluated for the presence of checkerboard artifact patterns. Checkerboard-style artifacts are visible in spectrograms as heavy, regularly repeating bars across the time axis while periodic broadband audio features are visible as lighter bars in the vertical direction along the frequency axis. An example of a non-aliased and aliased spectrogram can be seen in Figures 5.1 and 5.2, respectively.

Results and discussion of these experiments can be found in the next chapter.

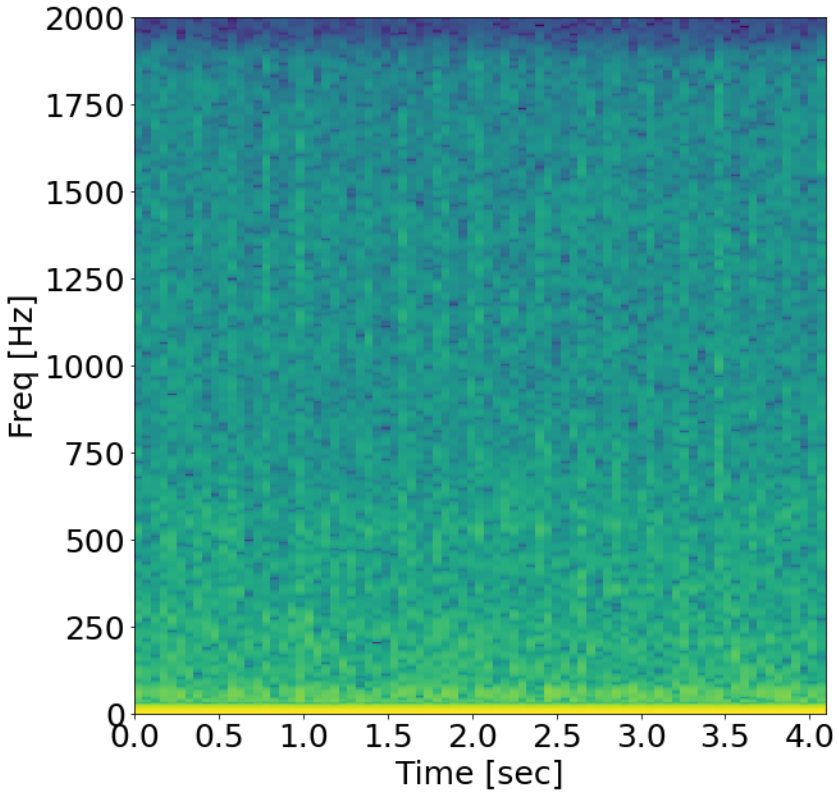

Figure 5.1. An example of a non-aliased spectrogram. Broadband audio features are visualized as brightly colored bars in the direction of the frequency axis.

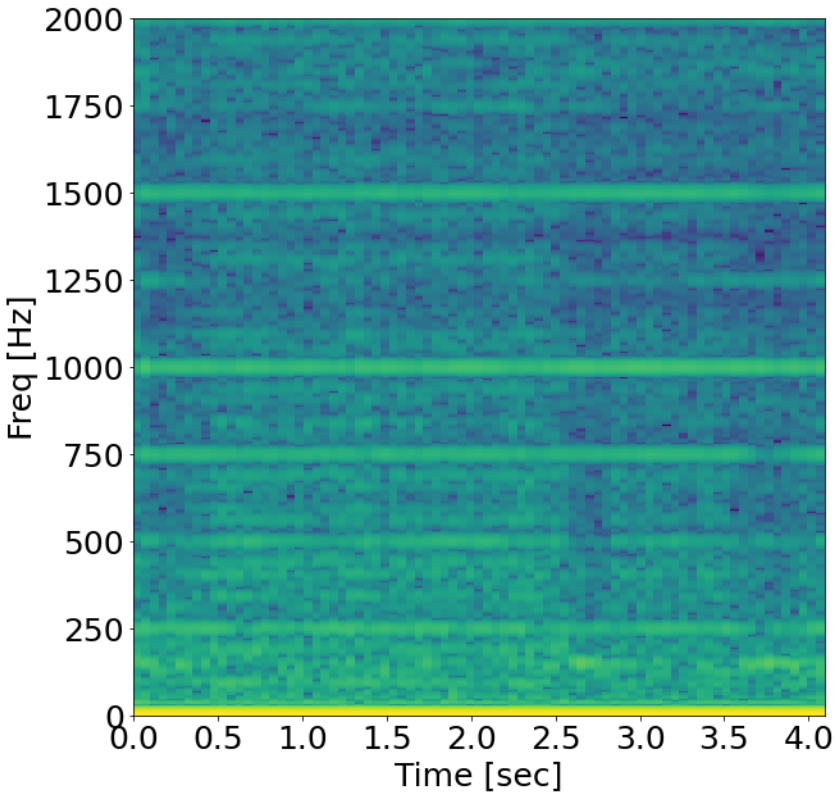

Figure 5.2. An example of a spectrogram showing checkerboard style aliasing artifacts. The artifacts are the strong repeating bars in in the direction of the time axis.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:
Results and Discussion

6.1 Architecture Exploration

Results from the architecture-exploration experiment are presented in Table 6.1. Two metrics are reported, Inception score (IS) and viability. As a reminder, an architecture was found to be viable if it had an Inception score greater than two and generated samples that could be audibly identified as a digit.

After 16 hours of training, the base WaveGAN architecture achieved an IS of 3.13 ± 0.02 and was deemed viable. The maximum reported WaveGAN IS of 4.67 ± 0.01 was achieved after three and a half day of additional training on a more powerful GPU [22]. Achieving an IS of 3.13 ± 0.02 after 16 hours shows that results indicating whether an architecture can produce sounds identifiable as a digit is possible with substantially less training. Apart from providing a baseline for architecture comparison, this validates the claim that speech-like audio is produced in the first few hours of training [22].

6.1.1 Decimation Filter Prototypes

Exploratory training on decimation filter prototypes (the binomial filter decimation, binomial filter decimation with mirror padding, and Hamming FIR decimation prototypes) all showed viable results indicating that the addition of decimation layers was a viable addition. Inception scores for decimation filter prototypes were improved up to 22% over the WaveGAN architecture for the same 16 hours of training. The addition of the decimation filter layers increased computational complexity of the discriminator resulting TangGAN training for fewer epochs in a constant amount of time. The most complex decimation layer, Hamming FIR Decimation, trained for less than half the number of epochs as WaveGAN, but still achieved a higher IS. The binomial filter with mirror padding achieved a slightly higher IS than the binomial filter without it, which is consistent with the hypothesis that mirror padding helps prevent edge artifacts leading to higher-quality sound generation. While the Hamming FIR Decimation prototype achieved a higher IS than WaveGAN, it did not

perform as well as the binomial filter decimation prototypes. This is likely the result of the Hamming FIR decimation prototype training for just over half the number of epochs that the binomial filter prototypes did and not indicative to the quality of the Hamming FIR filter in the decimation layer. With a larger and more stringent low-pass filter, additional training with the Hamming FIR decimation prototype would likely produce a better result than either of the binomial filter decimation prototypes. Since both Hamming FIR decimation and binomial filter were viable and showed IS improvement over WaveGAN, the Hamming FIR decimation prototype was chosen for further experimentation based on the better filter characteristics for decimation. In short, Hamming FIR filter had the greater potential to produce high-quality results.

6.1.2 Interpolation Prototypes

Architecture exploration with interpolation prototypes led to more ambiguous results. While every interpolation prototype produced viable architectures due to Inception scores higher than two, all Inception scores were lower than WaveGAN. Through subjective evaluation via listening, new architectures with interpolation were determined to have produced fewer artifacts and smoother sound compared to WaveGAN or any of the TangGAN decimation layer prototypes. Additionally, all of the interpolation prototypes were more computationally expensive than WaveGAN and trained fewer epochs despite training for 32 hours, or twice the computation time that WaveGAN trained for. The epochs trained by interpolation layers varied wildly with the complexity of the interpolation method. The fastest interpolation prototype, bilinear, trained nearly five times as many epochs as the slowest interpolation prototype, Lanczos-5.

The method of interpolation had little effect on the IS; all interpolation prototypes achieved Inception scores of about 2.30. Due to the limited amount of training time and Inception scores in this experiment, a determination of whether interpolation layer speed or complexity was more advantageous could not be made. For further experimentation, the bilinear interpolation and Lanczos-5 prototype were chosen for further experimentation as the fastest and most complex interpolation prototypes.

Table 6.1. Results from Architecture Exploration

Prototype Name	**Discriminator Architecture**	**Generator Architecture**	**Training Hours**	**Epochs**	**Inception Score**	**Viable?**
WaveGAN	WaveGAN with Phase Shuffle	WaveGAN	16	448	3.13 ± 0.02	Yes
Binomial Filter Decimation	Binomial Filter with Zero Pad	WaveGAN	16	345	3.81 ± 0.02	Yes
Binomial and Mirror Pad Decimation	Binomial Filter with Mirror Pad	WaveGAN	16	339	3.83 ± 0.03	Yes
Hamming FIR Decimation	Hamming FIR Decimation with Mirror Pad	WaveGAN	16	187	3.37 ± 0.02	Yes
Smoothing Filter Interpolation	Hamming FIR Decimation with Mirror Pad	Zero Order Hold and Smoothing Filter	32	291	2.30 ± 0.02	Yes
Bilinear Interpolation	Hamming FIR Decimation with Mirror Pad	Bilinear Interpolation	32	385	2.32 ± 0.01	Yes
Mitchell-Cubic Interpolation	Hamming FIR Decimation with Mirror Pad	Mitchell-Cubic Interpolation	32	110	2.27 ± 0.01	Yes
Lanczos Interpolation	Hamming FIR Decimation with Mirror Pad	Lanczos-5 Interpolation	32	78	2.25 ± 0.02	Yes

6.2 SC09 Generation

The results from training the WaveGAN, Hamming FIR decimation, bilinear interpolation, and Lanczos-5 interpolation architectures for one week can be seen in Table 6.2. Results from training WaveGAN, Hamming FIR decimation, and bilinear interpolation architectures for the same number of epochs can be seen in Table 6.3. Results were evaluated through Inception Score (IS) and Speech-to-Reverberation Modulation Energy Ratio (SRMR) calculated on $50,000$ generated samples.

6.2.1 Constant Training Time

After a week of training the WaveGAN architecture achieved an IS of 4.25 ± 0.05 which is comparable to the best published results of 4.67 ± 0.01 [22]. The Hamming FIR decimation architecture achieved a higher inception score than WaveGAN despite for training less than half the number of epochs. To our surprise, training for a week (168 hours) instead of 36 hours produced only a modest increase in IS for the bilinear interpolation architecture and no increase at all for the Lanczos-5 interpolation architecture

The low Inception scores for the interpolation architectures contrasted with qualitative listening evaluations of the generated samples. The generated samples via interpolation sounded more intelligible and natural than other samples implying that they contained less artifacts. In order to quantify this finding, the non-intrusive, wideband SRMR metric designed to measure sound distortion due to artifacts was used [40]. The less artifact distortion in a sound, the higher the SRMR score is. The highest SRMR was achieved by the TangGAN bilinear interpolation architecture while the lowest SRMR was achieved by WaveGAN indicating the the bilinear interpolation architecture produced the lowest amount of artifact distortion. All TangGAN architectures improved SRMR over WaveGAN.

This experiment informed tradeoffs between interpolation method quality and speed. The bilinear interpolation architecture trained over four times as many epochs in one week and achieved a higher IS and SRMR than the Lanczos-5 interpolation method. While the bilinear interpolation architecture trained at 87% of the speed of the Hamming FIR decimation architecture, the Lanczos-5 method only trained at 20%. Because the Lanczos-5 interpolation architecture trained over ten times slower than the WaveGAN architecture, it was omitted from the constant time SC09 digit generation experiment. Training the

Table 6.2. Results from SC09 Constant Time Generation

Architecture	Epochs	Inception Score	SRMR
SC09 (Training Data)	—	9.18 ± 0.04	6.34
WaveGAN	5317	4.25 ± 0.05	4.28
Hamming FIR Decimation	2309	4.37 ± 0.03	4.37
Bilinear Interpolation	2010	2.50 ± 0.02	4.90
Lanczos Interpolation	452	2.25 ± 0.02	4.64

TangGAN Lanczos-5 architecture to the same number of epochs that WaveGAN trained in one week would take over ten weeks.

6.2.2 Constant Training Epochs

In one week, the WaveGAN architecture trained for 5317 epochs. It took the TangGAN Hamming FIR decimation architecture two weeks and two days and the TangGAN bilinear interpolation architecture two weeks and four and a half days to reach the same number of epochs.

Training the TangGAN Hamming FIR decimation and TangGAN bilinear interpolation architectures led to a nearly 20% increase in IS for the bilinear interpolation architecture, but no statistically significant increase for the TangGAN Hamming FIR decimation architecture. Unexpectedly, the SRMR for both TangGAN architectures went down significantly to below WaveGAN's SRMR. Examining plots of loss functions for the WaveGAN, Hamming FIR decimation, and bilinear interpolation architectures in Figures 6.1, 6.2, and 6.3 may

Table 6.3. Results from SC09 Constant Epochs Generation

Architecture	Epochs	Time (hours)	Inception Score	SRMR
SC09 (Training Data)	—	—	9.18 ± 0.04	6.34
WaveGAN	5317	168	4.25 ± 0.05	4.28
Hamming FIR Decimation	5317	390	4.40 ± 0.02	4.10
Bilinear Interpolation	5317	447	2.69 ± 0.02	4.04

provide an explanation of this behavior. After epoch 1500 for the WaveGAN architecture the generator and discriminator loss functions fall into stable states centered around 0.0 and -5.5 respectively. The loss functions for the Hamming FIR Decimation architecture never reach a steady state. While smooth, the discriminator loss function has a negative gradient after epoch 600. The generator loss function behaves erratically between epoch 1200 and 3300 after which it increases at a near constant rate of change. The diverging discriminator and generator loss functions may indicate that performance of this architecture will continue to get worse the longer it is trained. Perplexingly, the generator loss function behavior for the Hamming FIR decimation architecture is so different from WaveGAN's generator loss function despite being the same generator. The difference in behavior must therefore be caused by the discriminator influencing the generator through the generator's loss function. Finally, the loss functions for the bilinear interpolation architecture demonstrate a third sort of behavior. Like the Hamming FIR decimation filter, the bilinear interpolation generator loss function has a gentle negative slope after epoch 600 even though the generator architectures are different. The discriminator, however, exhibits different behavior even though the architectures are the same between the Hamming FIR decimation and bilinear interpolation architectures. Overall, the generator loss function seems to exhibit minimal growth punctuated with a short period of intense growth from epoch 2250 to 2600 followed by increased instability. The generator and discriminator loss functions seem to increasingly diverge after epoch 4500. While correlation between loss functions and generation performance is not well understood, the introduction of anti-aliasing filters and interpolation layers seem to have a drastic effect on generator loss functions.

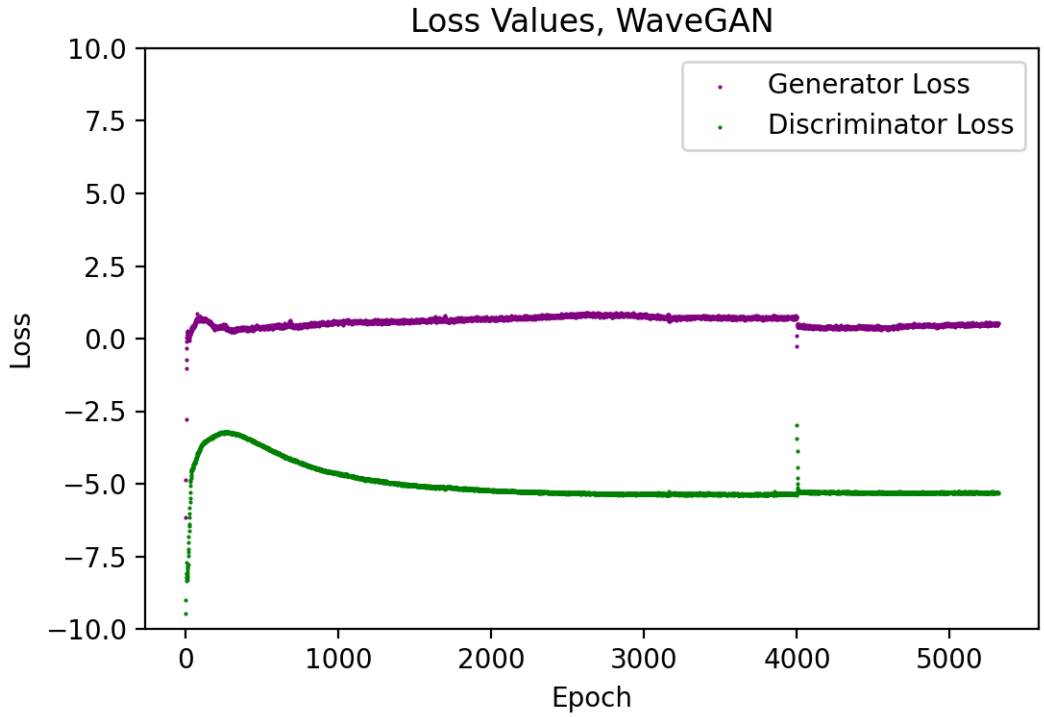

Figure 6.1. Loss functions for the WaveGAN discriminator and generator over 5317 epochs. Notice how the generator loss function quickly reaches a value of about 0.0 and the discriminator reaches a value of about -5.5 and does not deviate as the architecture trains. The WaveGAN architecture loss functions exhibit a stable behavior with IS and SRMR scores rising as WaveGAN trains. The anomalous loss points at epoch 4000 are due to a reset of the WaveGAN optimizer, ADAM, caused by an interruption of training. After a few epochs ADAM warms up and the stable behavior of the WaveGAN loss functions continues.

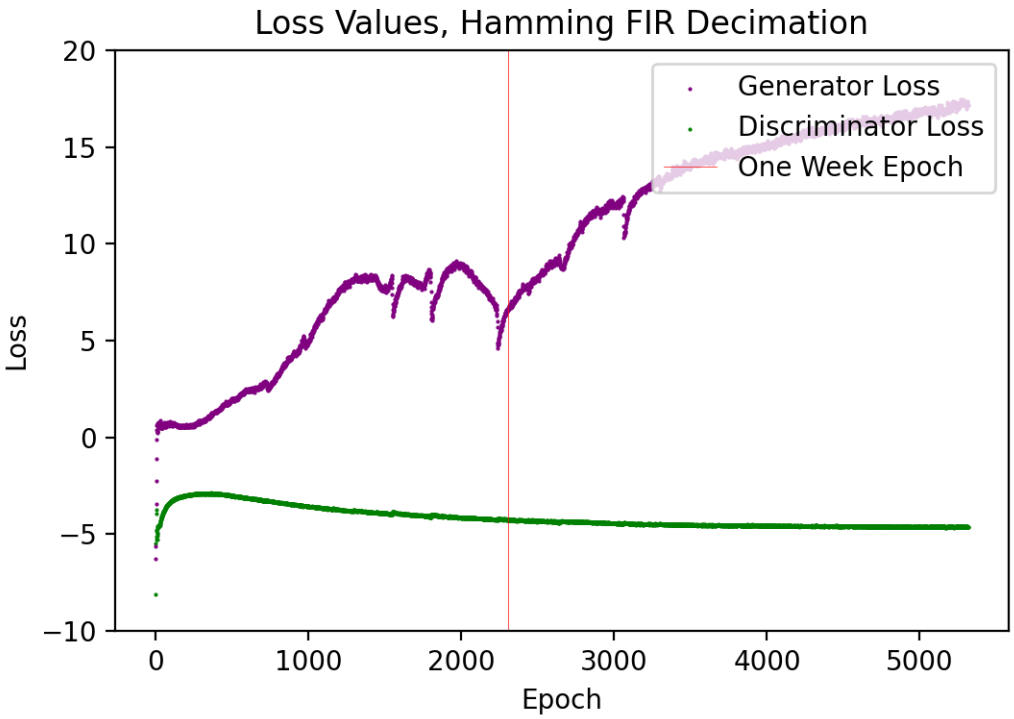

Figure 6.2. Loss functions for the TangGAN Hamming FIR Decimation discriminator and generator over 5317 epochs. The discriminator loss is well behaved while the generator loss is not. The generator loss exhibits different behavior from the WaveGAN architecture despite having the same generators. As the discriminator and generator losses diverged between epoch 2309 and 5317, IS and SRMR went down.

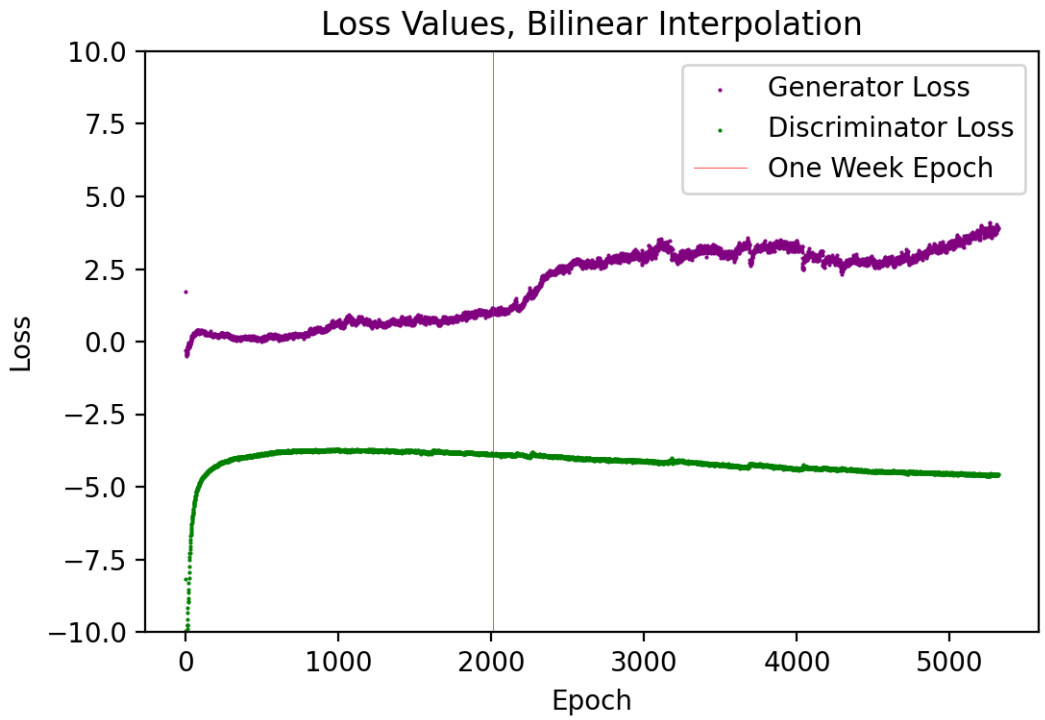

Figure 6.3. Loss functions for the TangGAN bilinear interpolation discriminator and generator over 5317 epochs. The generator and discriminator losses seem to achieve a state of relative stability until about epoch 2250, after which they increasingly diverge. SRMR decreased and IS increased between epoch 2010 and epoch 5317.

6.3 Pure Tone Generation

Table 6.4 shows the results of 250 epochs of training on the pure tone dataset by the WaveGAN, Hamming FIR Decimation, and Bilinear Interpolation training methods. This experiment calculated the average Total Harmonic Distortion (THD) and signal-to-noise ratio (SNR) of $50,000$ generated samples to quantitatively evaluate artifacts introduced by the tested GAN architectures.

In the theoretical case up to the quantization level, a pure tone has a THD of $-\infty$ and a SNR of ∞ because there are no harmonic distortions or noise to compare the signal to. By training on architectures on a pure tone data set, the introduction of any artifacts would cause finite THD and SNR values. The Hamming FIR decimation architecture improves THD by over a decibel and a half and SNR by four decibels over the WaveGAN architecture suggesting that the Hamming FIR decimation architecture generates less artifacts. Concerningly, the bilinear interpolation architecture has the highest THD and lowest SNR in direct contradiction to the SRMR results from the SC09 constant time generation experiment.

Table 6.4. THD and SNR for Pure Tone Generation

Architecture	**Epochs**	**THD (dB)**	**SNR (dB)**
Pure Tone	—	$-\infty$	∞
WaveGAN	250	−38.58	25.37
Hamming FIR Decimation	250	−40.82	29.59
Bilinear Interpolation	250	−36.75	13.77

6.4 Underwater Acoustic Proof-of-Concept

Sounds generated from training on the underwater acoustic dataset were used to create spectrograms. These spectrograms were qualitatively evaluated for signs of checkerboard aliasing. The generated sounds were also listened to in order to ensure they generated realistic underwater sounds.

As seen in Figure 6.4 spectrograms generated from real audio show a smooth and uniform pattern, especially in the frequency domain. In general, signal intensity is highest in the lower frequencies and lowest in the higher frequencies which is correlates with higher

frequency sounds being attenuated more rapidly underwater. The very high intensity signal below 25 Hz is likely due to instrument noise on the hydrophone that recorded the dataset. Audio features from ship noises, such as propeller blade rate or engine revolutions are present in these spectrograms as repeating vertical lines across many frequencies as they are broad band sources of noise. Largely absent from the real audio spectrograms are horizontal bars indicating monotonal sources of energy.

Figure 6.5 shows spectograms created by the WaveGAN architecture. In contrast to the real audio spectrograms, the ones generated by the WaveGAN architecture have prominent horizontal bars often spaced at regular intervals. These regularly spaced monotonal energy bands are consistent with how checkerboard-style artifacts appear in spectrograms. The monotonal energy bands almost always appear with greater intensity than the audio features in the sounds meaning artifacts constitute the defining characteristics of the WaveGAN generated sounds.

Spectrograms generated by the TangGAN Hamming FIR decimation architecture are visible in 6.6. Monotonal energy bands are much less prevalent in both quantity and intensity than spectrograms from the WaveGAN architecture. Audio features are weaker than they are in the real data audiograms but more discernible than in the WaveGAN architecture due to the drastic reduction of monotonal energy bands. Many spectrograms feature the strongest artifact at $1,000$ Hz. While the exact cause of this artifact is not known, $1,000$ Hz is one quarter of the sample rate and in TangGAN and both the decimation factor in the discriminator and growth factor in the generator are four making the artifact at $1,000$ Hz likely related to these processes.

The final set of spectrograms generated by the TangGAN bilinear interpolation architecture can be seen in Figure 6.7. These spectrograms show no high-intensity checkerboard-style artifacts. Audio features are bright and easy to observe. All of the spectrograms display an artifact at $1,000$ Hz, but unlike the artifacts generated by the WaveGAN and bilinear interpolation architectures, the bilinear interpolation architecture artifact is characterized by a suppression of signal strength at $1,000$ Hz instead of an excess. An artifact that suppresses signal is preferable to one that creates a signal excess because an signal excess masks other signals present in a sound when listened to. Like the $1,000$ Hz artifact in the Hamming FIR decimation architecture, the signal suppression artifact produced by the

bilinear interpolation architecture is likely related to the growth and decimation factors in TangGAN.

Although just a proof-of-concept that TangGAN can be used to generate underwater sounds, this experiment provides a visual way to demonstrate the impact reduction of artifacts has on the quality of generated sounds.

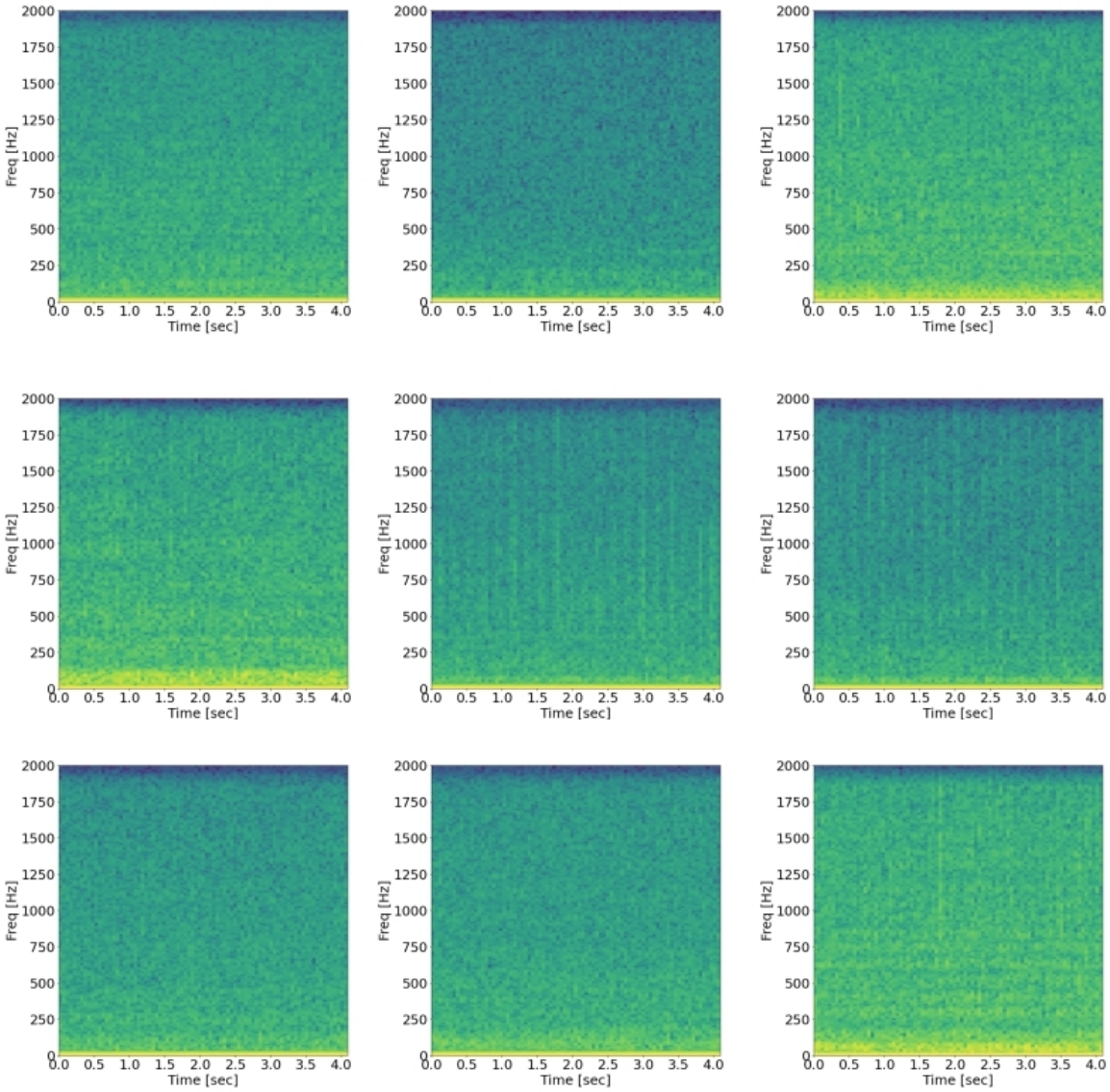

Figure 6.4. These are spectrograms generated from real recordings of ships greater than $4,000$ tons. Portions of the spectrum that have more energy are more brightly colored. Ship sounds make a broadband periodic signal which can be seen as repeating vertical bars in the spectrograms. The middle spectrogram is an example of a recording with an especially strong ship audio.

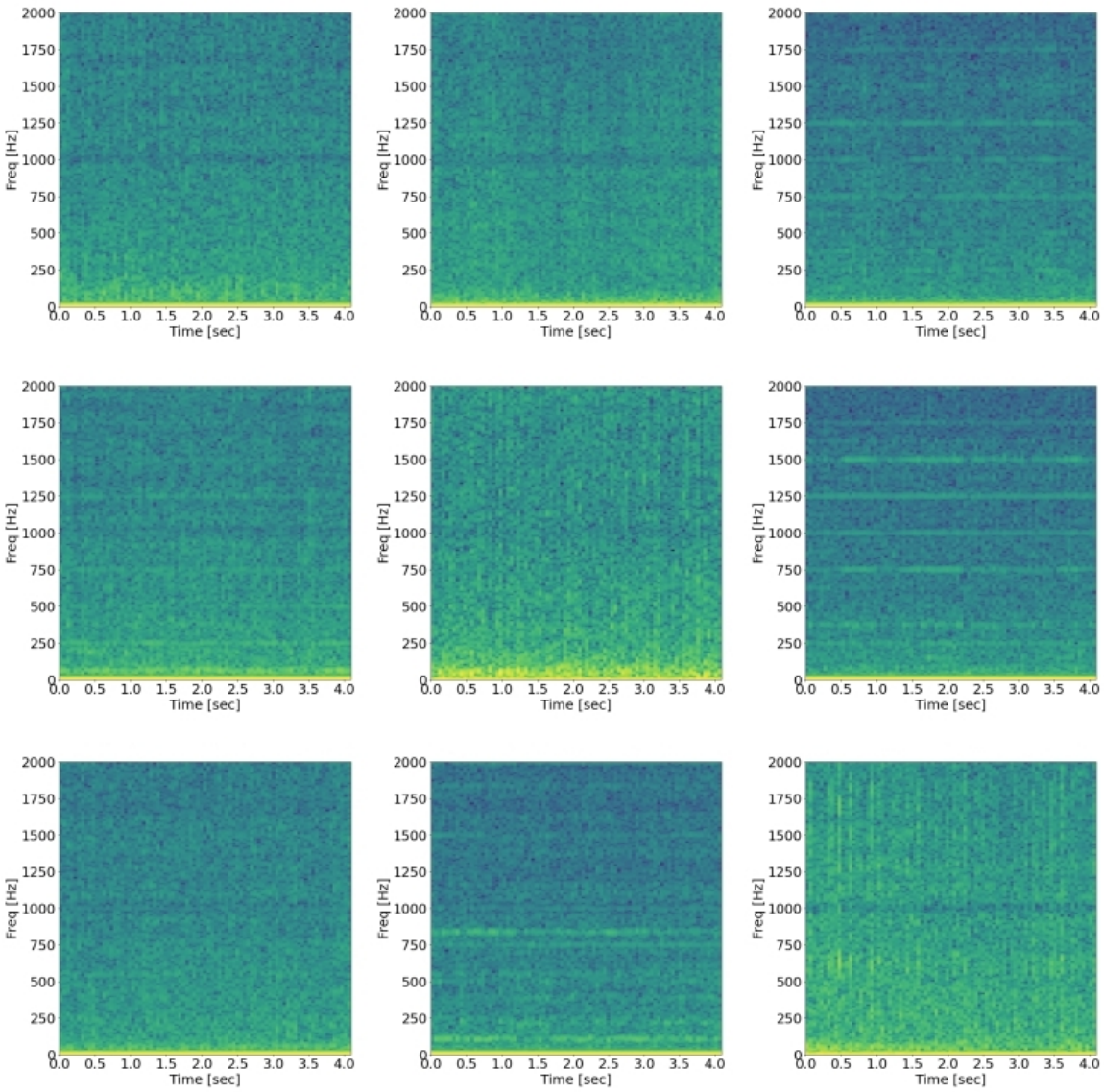

Figure 6.5. Spectrograms generated from the WaveGAN architecture after training for one week on the underwater acoustics dataset. Notice how many of the spectrograms have high-intensity horizontal bars indicating monotonal signals that are not present in the spectrogams. The monotonal signals often occur at regular intervals indicating that they are checkerboard-style artifacts. The middle right spectogram shows an sample with particularly severe checkerboard artifacts.

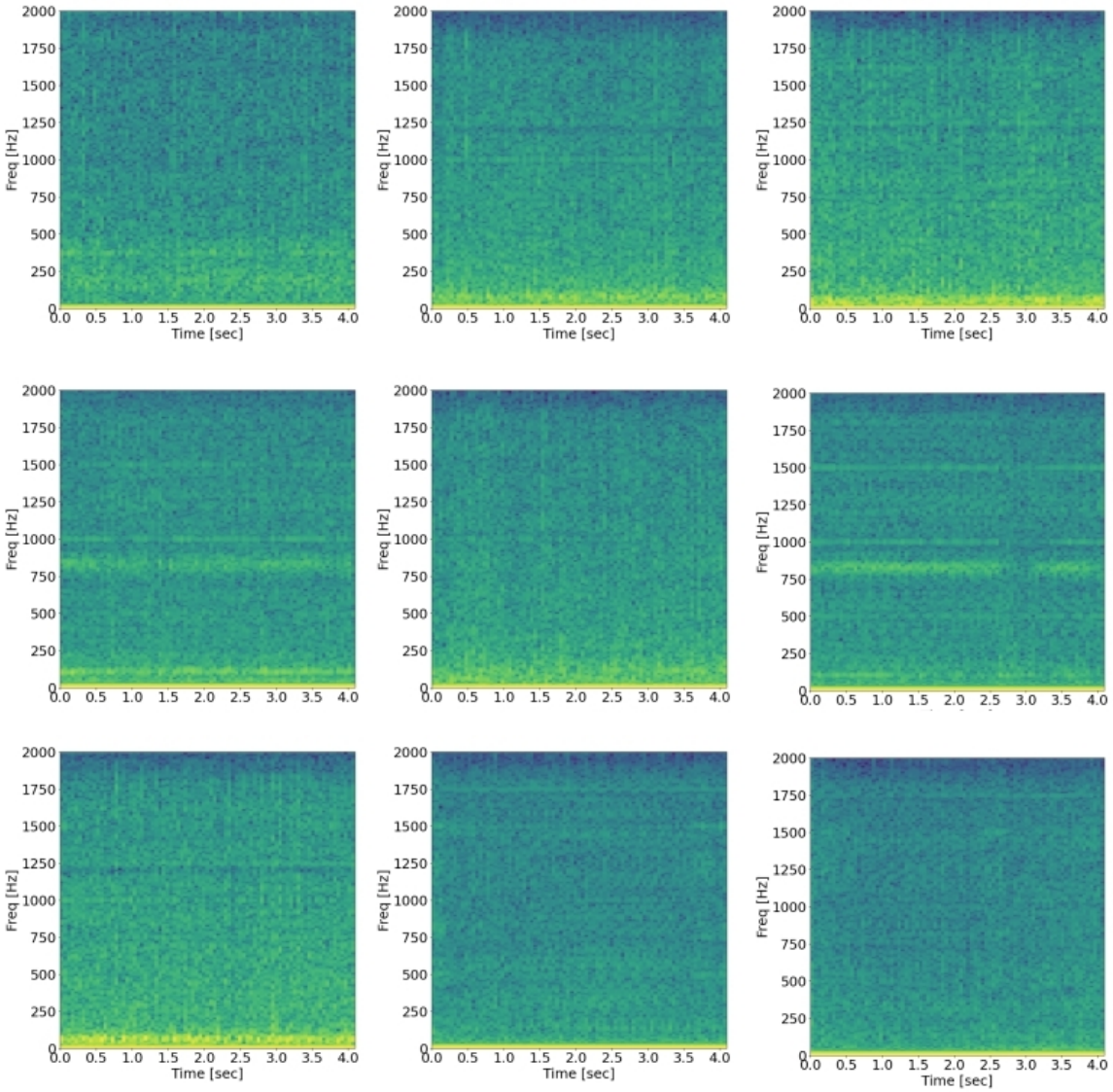

Figure 6.6. These spectrograms were generated from samples created by the TangGAN Hamming FIR Decimation filter prototype. The monotonal signals from checkerboard artifacts are less prevalent than they are in WaveGAN generated samples. Signals from ship sounds are easier to identify because they do not have strong artifacts overlaying them.

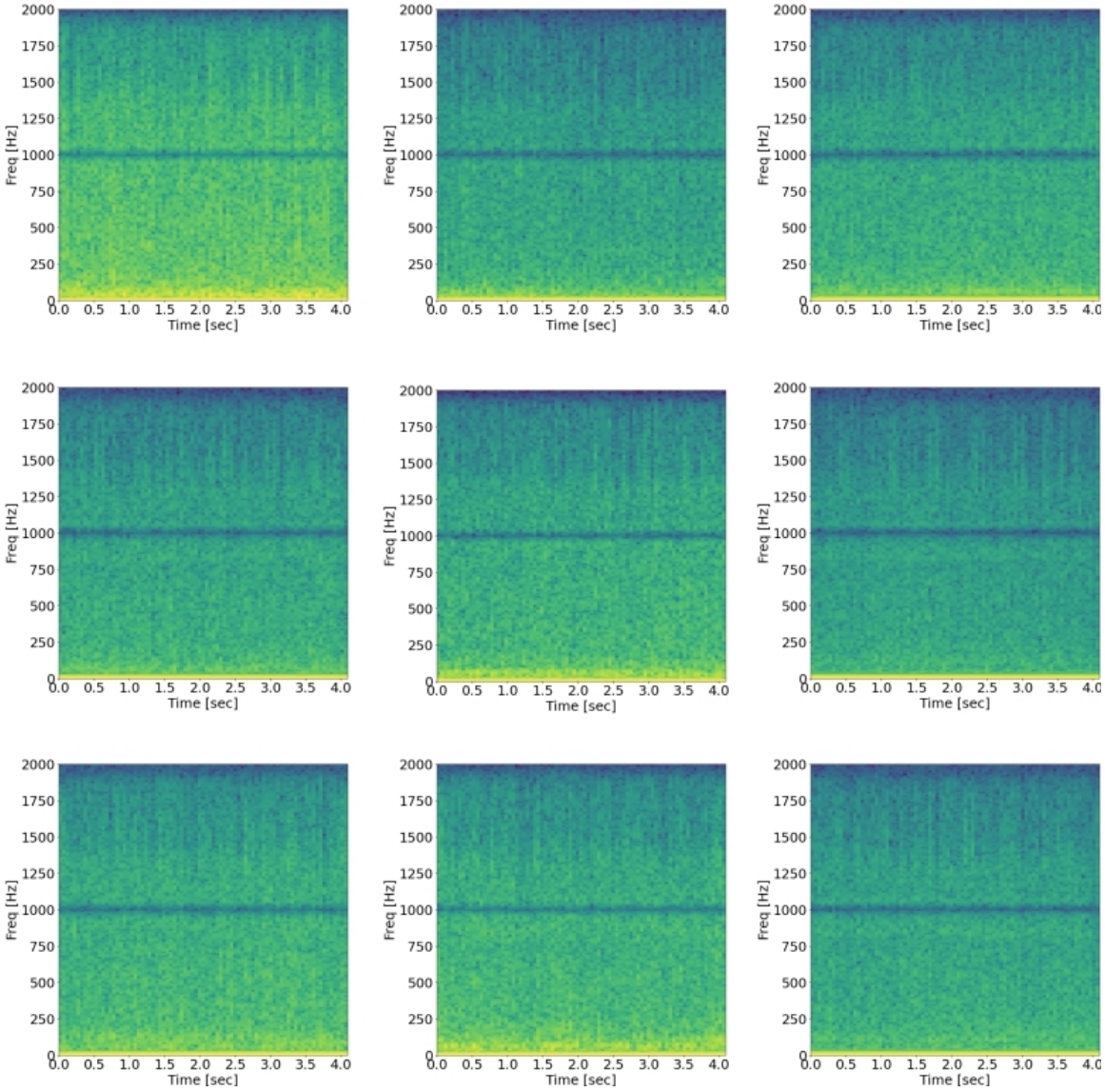

Figure 6.7. Spectrograms from samples created by the TangGAN bilinear interpolation architecture. Monotonal signals artifacts have been completely eliminated in these spectrograms. All the samples feature an artifact of unknown origin at $1,000$ Hz, but as this a signal suppression artifact instead of a signal excess artifact meaning it is not as noticeable to the human ear. Ship audio features, represented by vertical bars, are more visible than in the WaveGAN or Hamming FIR Decimation architecture samples.

CHAPTER 7:
Conclusion and Future Work

7.1 Conclusion

The main contribution of this thesis is an introduction of a novel method for the generation of audio, called TangGAN, by introducing decimation and interpolation layers into the discriminator and generator of a GAN architecture. The decimation layers prevent aliasing artifacts in the discriminator. The interpolation layers prevent aliasing in the generator while allowing convolutional layers to focus on learning audio features. The introduction of interpolation and decimation layers in an audio GAN architecture reduced artifacts in generated audio and improved the quality of sound as measured by several metrics.

This thesis introduces new metrics to identify parasitic artifacts in outputs of GAN architectures for audio generation. Total harmonic distortion (THD) and signal-to-noise ratio (SNR) which are traditionally used in signal processing to identify signal quality were used here to quantify artifacts introduced to single tone samples generated by GAN architectures. Speech-to-reverberation modulation energy ratio (SRMR), a non-intrusive speech quality metric designed to measure distortions in speech audio introduced by dereverberation algorithms, was used to measure the quality of sound produced by GAN architectures. Since SRMR correlates well with mean opinion scores (MOS) [40] the adoption of SRMR for audio GAN research helps alleviate the need for expensive human trials to measure sample quality.

These new metrics were combined with more established ones to compare various architectures of TangGAN with the current state-of-the-art WaveGAN. TangGAN architectures and WaveGAN were compared across three different datasets: spoken digits in the SC09 dataset; single frequency tones in the pure tone dataset; and ship noises on the underwater acoustics dataset. The best samples produced by TangGAN architectures produced higher SRMR than WaveGAN, with the TangGAN bilinear interpolation architecture providing the highest SRMR overall indicating that TangGAN produced higher quality, lower artifact audio than WaveGAN. While WaveGAN achieved a higher Inception score (IS) than the TangGAN

bilinear interpolation architecture, the TangGAN Hamming FIR architecture outperformed WaveGAN in both IS and SRMR. On the pure tone dataset, the TangGAN Hamming FIR architecture showed significant improvement over WaveGAN in both THD and SNR, once again demonstrating TangGAN's capability to improve audio quality through the reduction of artifacts. Finally, after training on the underwater acoustic sound set, spectrograms qualitatively showed TangGAN reduces artifacts present in generated samples resulting in improved audio quality.

Research into generating realistic audio is not a field that is limited to the academic and consumer market interest. Submarines rely on stealth to keep their crews safe. Below the waves, underwater communications have to be carried out via acoustic waves; light and radio frequencies are simply attenuated too quickly. The problem with acoustic communications is that they can be intercepted by adversarial forces. Secure communications would allow submarines to increase their effectiveness whether by enabling submarines to share contact reports and coordinate in a wolf pack or to share sensor information with and issue commands to underwater drones. A realistic audio generator can provide a context-aware system to produce sounds indistinguishable from environmental sounds for these communications to be hidden in via steganography. Communications that become part of the environment no longer reveal units' positions. Realistic audio generation can also be used to create signals that conceal or confound sensors from identifying targets by disguising friendly units' audio signatures. Just like visual camouflage makes units harder to find on the surface, audio camouflage can make units harder to find underwater increasing the safety of friendly units. Furthermore, realistic audio generation is being researched by our competitors. Take [19] for example where researchers funded by the government of the People's Republic of China are conducting research on expanding datasets of "hard to collect" underwater sources in order to improve automatic audio classifiers. Or [45] where researchers once again funded by the People's Republic of China developed a GAN to embed communications in a carrier sound without audibly changing the carrier sound. This thesis focuses on generating realistic acoustic samples. We hope that it stimulates further interest in this field because audio generation can help the United States Navy increase the connectivity and survivability of its fleet.

7.2 Future Work

TangGAN has made significant contributions to the field of computer-audio generation validated by TangGAN's performance compared to the state-of-the-art WaveGAN. That being said, more work can be done to improve TangGAN. Some suggestions follow.

First, while TangGAN can produce audio that exceeds WaveGAN's in the same amount of training time, it still trains for a week on the SC09 dataset. Two methods show promise for reducing the amount of training time it takes TangGAN to produce high-quality audio. The first method is to implement parallelization. Parallelization would allow TangGAN to train on multiple GPUs at once. The NVIDIA team in [10] significantly cut down the amount of time it took for their GAN to train by conducting training on eight GPUs. The other method that may help TangGAN train faster is to progressively grow TangGAN as described in [10] and discussed in Chapter 2. Progressively growing TangGAN would allow TangGAN to initially train on a smaller architecture to learn low-resolution features before increasing architecture size to learn higher resolution features. Progressively growing GAN architecture allowed NVIDIA to train its GAN over five times faster than a non-progressively growing version. Progressively growing a GAN would be especially helpful if it were desired to have TangGAN generate samples with more than $16,384$ time steps because the addition of time steps require additional layers and weights to produce. Progressively growing TangGAN can help avoid the computational cost of additional weights by only adding them in the final stages of training. Feature size is something that must be given special attention in making a progressively growing version of TangGAN. While an audio sample produced by TangGAN might be roughly the same size as an image produced by NVIDIA's progressively growing GAN, the feature sizes in audio tend to be larger and periodically repeat more than visual features. This means the architecture of NVIDIA's progressively growing GAN would need to be modified for TangGAN.

An alternate method to increase TangGAN's training speed is to use transfer learning. Transfer learning on TangGAN may or may not be appropriate between datasets that have very different audio features such as a dataset of spoken language and one of underwater ship noises. On classes that have similar audio features, however, such as a dataset of ship sounds from vessels over $5,000$ tons and ship sounds from vessels under $5,000$ tons transfer learning may be more successful. The features learned from training on one of the datasets may be easily converted into features from the other dataset by loading the weights from

the first dataset and training for a shorter number of epochs on the second dataset. The potential benefits of using transfer learning to quickly produce sounds from similar classes would be especially useful for the generation of labeled datasets for training classifiers and merits further research.

Another field for further investigation is to determine why both the TangGAN Hamming FIR decimation and bilinear interpolation architectures produced artifacts at $1,000$ Hz, exactly one quarter of the sampling rate. Oddly, the artifact was the introduction of erroneous signal strength for the Hamming FIR decimation prototype and signal suppression for the bilinear interpolation prototype. While it seems likely that the artifacts were introduced at $1,000$ Hz because of the decimation and growth factors in the discriminator and generator were both 4, why they were introduced and why they are different types of artifacts requires further investigation. While the bilinear interpolation architecture's suppression artifact may not be noticeable by the human listener, both artifacts would be easily detectable through spectrographic analysis or by a neural network classifier and would need to be dealt with prior to generating sounds that would be truly indistinguishable from real sounds.

A vital hyperparameter that was not extensively explored during the architecture exploration were filter and kernel sizes. In the current TangGAN discriminator architecture the decimation filter and convolutional kernel are both set to be the same size. The same applies for the interpolation layer kernel and convolutional layer sizes. This does not have to be the case and tying the filter and kernel sizes together may actually be a source of edge artifacts especially in cases with large filter and kernel sizes with small sample sizes. For example, the last decimation and convolutional layers in the Hamming FIR decimation discriminator architecture have a size of 31 and are applied to a sample only 64 time steps long. The decimation layer pads the input via mirror padding with 32 samples on each side of the input sample. The convolutional layer has a similar functionality except Tensorflow automatically applies zero-padding. With a convolutional kernel size of 31, Tensorflow pads the size 64 sample with 30 zeroes to produce an output of the same size of 64 samples. In other words, almost a third of the signal that is convolved over consists of zeroes from padding and this may cause the addition of edge artifacts. Conversely, if a very efficient low-order filter of say size 7 could be found, having a convolutional kernel size of 7 would almost certainly be too small to learn audio features [22]. Another option to produce anti-aliasing while minimizing edge artifacts that is not possible to implement with a same-sized decimation

filter and covolutional kernel size would be to stack three filtering operations of lower order filters one after another. Being able to separate the decimation and interpolation filter sizes from convolutional kernel sizes would give greater flexibility in the varieties of decimation and interpolation layers that could be explored.

A further avenue for research would be in trying to understand how loss functions affect audio quality. After training TangGAN architectures to match the number of epochs that WaveGAN trained for in one week, the quality of sounds as measured by both IS and SRMR went down. The TangGAN generator and discriminator losses behaved much differently than the WaveGAN losses which quickly reached a stable state. Instead, for both TanGAN architectures, the generator loss function behaved erratically and diverged increasingly rapidly from the discriminator loss function as training progressed. Experimentation determined that audio from the point of minimum divergence between loss functions did not produce superior quality audio. There is likely correlation between the generator-discriminator loss function divergence and audio quality which requires further investigation. Finding this relation could allow for the development of early stopping criteria during training or a method of determining when TangGAN produced an optimally trained generator without having to create data from every saved checkpoint.

Finally, to our knowledge, no work has been done on what audio features an audio GAN learns. By deconstructing the convolutional layers in the discriminator and querying the learned features, not only could analysis of learned features help determine how to build better discriminators by giving insight into the optimal audio convolutional layer kernel size it could also help develop better audio classifiers. Deconstructing the convolutional layer in the generator would give insight into how audio features are grown from low-resolution to high-resolution as samples progress through the GAN.

TangGAN is a robust audio generation architecture already, but it can be made better. We hope the foregoing suggestions help inspire further research in the field of audio generation because there is much left to discover.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] Naval Historical Society of Australia, "British and German submarine statisitics of World War II," https://www.navyhistory.org.au/british-and-german-submarine-statistics-of-world-war-ii/, December 1972.

[2] N. Thiem, "TangGAN Code," 2020. Available: https://gitlab.nps.edu/student_thesis_orescanin/tanggan

[3] N. Thiem, "TangGAN Training Data and Weights," 2020. Available: https://nps.app.box.com/folder/121487361329

[4] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems*, 2014, pp. 2672–2680.

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[6] J. M. Susskind, A. K. Anderson, and G. E. Hinton, "The Toronto face database," *Department of Computer Science,* University of Toronto, Toronto, ON, Canada, Tech. Rep., 3, 2010.

[7] A. Krizhevsky, "Learning multiple layers of features from tiny images," *Department of Computer Science,* University of Toronto, Toronto, ON, Canada, Tech. Rep, April 2009.

[8] Y. Bengio, G. Mesnil, Y. Dauphin, and S. Rifai, "Better mixing via deep representations," in *International Conference on Machine Learning*, 2013, pp. 552–560.

[9] Y. Bengio, E. Laufer, G. Alain, and J. Yosinski, "Deep generative stochastic networks trainable by backprop," in *International Conference on Machine Learning*, 2014, pp. 226–234.

[10] T. Karras, T. Aila, S. Laine, and J. Lehtinen, "Progressive growing of GANs for improved quality, stability, and variation," *arXiv e-prints arXiv:1710.10196*, Oct 2017.

[11] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[12] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint*, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[14] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "LSUN: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[15] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3730–3738.

[16] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of StyleGAN," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 8110–8119.

[17] J. West and C. Bergstrom, "Which face is real?" 2019. Available: www.whichfaceisreal.com

[18] G. L. Grinblat, L. C. Uzal, and P. M. Granitto, "Class-splitting generative adversarial networks," *arXiv preprint arXiv:1709.07359*, September 2017.

[19] F. Liu, Q. Song, and G. Jin, "Expansion of restricted sample for underwater acoustic signal based on generative adversarial networks," *Proceedings of SPIE*, vol. 11069, no. 1106948, pp. 1–8, May 2019.

[20] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[21] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[22] C. Donahue, J. McAuley, and M. Puckette, "Adversarial audio synthesis," in *International Conference on Learning Representations*, 2018.

[23] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[24] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Advances in Neural Information Processing Systems*, 2017, pp. 5767–5777.

[25] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," *arXiv preprint arXiv:1804.03209*, April 2018.

[26] J. Engel, K. K. Agrawal, S. Chen, I. Gulrajani, C. Donahue, and A. Roberts, "GANSynth: Adversarial neural audio synthesis," in *International Conference on Learning Representations*, 2018.

[27] A. Oord, Y. Li, I. Babuschkin, K. Simonyan, O. Vinyals, K. Kavukcuoglu, G. Driessche, E. Lockhart, L. Cobo, F. Stimberg *et al.*, "Parallel wavenet: Fast high-fidelity speech synthesis," in *International Conference on Machine Learning*, 2018, pp. 3918–3926.

[28] Xcelerit, "Benchmarks: Deep learning nvidia P100 vs. V100 gpu," https://www.xcelerit.com/computing-benchmarks/insights/benchmarks-deep-learning-nvidia-p100-vs-v100-gpu/, November 2017.

[29] R. Zhang, "Making convolutional networks shift-invariant again," in *International Conference on Machine Learning*, 2019, pp. 7324–7334.

[30] Creative Commons, "Creative commons license attribution 4.0 international (CC BY 4.0)," 2018. Available: https://creativecommons.org/licenses/by/4.0/

[31] I. S. Pigeon, "The non-linearities of the human ear," 2018. Available: https://www.audiocheck.net/soundtests_nonlinear.php

[32] Google, "TensorFlow API documentation," https://www.tensorflow.org/api_docs, August 2020.

[33] T. Margolina, private communication, July 2020.

[34] A. Pfau, private communication, August 2020.

[35] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*, 3rd ed. Upper Saddle River, N.J: Prentice Hall, 1996.

[36] A. V. Oppenheim, *Discrete-Time Signal Processing*, 2nd ed. Upper Saddle River, N.J: Prentice Hall, 1999.

[37] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Advances in Neural Information Processing Systems*, 2016, pp. 2234–2242.

[38] International Telecommunication Union, "Wideband Extenstion to Rec. P.862 for the Assesement of Wideband Telephone Networks and Speech Codecs," International Telecommunication Union, Tech. Rep., November 2007. Available: https://www.itu.int/rec/T-REC-P.862.2

[39] L. Malfait, J. Berger, and M. Kastner, "P.563-the ITU-T standard for single-ended speech quality assessment," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 14, no. 6, pp. 1924–1934, 2006.

[40] T. H. Falk, C. Zheng, and W.-Y. Chan, "A non-intrusive quality and intelligibility measure of reverberant and dereverberated speech," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 7, pp. 1766–1774, 2010.

[41] P. C. Loizou, *Speech Enhancement: Theory and Practice*. CRC press, 2013.

[42] J. F. Santos, T. H. Falk, C. Zheng, W.-Y. Chan, and M. Senoussaoui, "SRMR toolbox," 2016. Available: https://github.com/MuSAELab/SRMRToolbox

[43] D. Shmilovitz, "On the definition of total harmonic distortion and its effect on measurement interpretation," *IEEE Transactions on Power Delivery*, vol. 20, no. 1, pp. 526–528, 2005.

[44] "MATLAB signal processing toolbox," 2020, MathWorks, Natick, MA, USA.

[45] D. Ye, S. Jiang, and J. Huang, "Heard more than heard: An audio steganography method based on GAN," *arXiv preprint arXiv:1907.04986*, 2019.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California